

# Deploying a Cloud on IBM System z

Understanding cloud component models

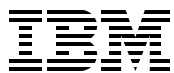
Configuring z/VM and Tivoli Service Automation Manager

Deploying the cloud with a step-by-step checklist



Mike Buzzetti  
James Kuchler  
Charlie Lawrence





International Technical Support Organization

**Deploying a Cloud on IBM System z**

February 2011

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (February 2011)**

This edition applies to Version 7.2.0.\* of Tivoli Service Automation Manager and z/VM 5.4

This document created or updated on May 2, 2011.

**© Copyright International Business Machines Corporation 2011. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

|   |      |
|---|------|
| <b>Notices</b> .....  | v    |
| Trademarks .....  | vi   |
| <b>Preface</b> .....  | vii  |
| The team who wrote this paper .....   | vii  |
| Now you can become a published author, too! .....                                     | viii |
| Comments welcome .....  | ix   |
| Stay connected to IBM Redbooks .....  | ix   |
| <b>Chapter 1. Introduction</b> .....  | 1    |
| 1.1 The Managed From, Through, and To Component Models .....                          | 3    |
| 1.1.1 The Managed From component model .....  | 4    |
| 1.1.2 The Managed Through component model .....                                       | 5    |
| 1.1.3 The Managed To component model .....  | 6    |
| 1.2 Summary .....   | 6    |
| <b>Chapter 2. Configuring Managed Through (z/VM)</b> .....                            | 7    |
| 2.1 Overview of z/VM configuration .....  | 8    |
| 2.2 Details of the Managed Through configuration .....                                | 8    |
| 2.2.1 Setting up the SYSTEM CONFIG file for DASD .....                                | 9    |
| 2.2.2 Setting up the SYSTEM CONFIG file for networking .....                          | 11   |
| 2.2.3 Specifying TCP/IP devices .....   | 13   |
| 2.2.4 Updating additional TCP/IP settings .....                                       | 14   |
| 2.2.5 SYSTEM DTCPARMS relative to TCP/IP stack .....                                  | 15   |
| 2.2.6 Configuring DIRMAINT .....  | 15   |
| 2.2.7 Setting up the EXTENT CONTROL FILE .....  | 17   |
| 2.2.8 Creating MAPSRV and MAPAUTH .....   | 18   |
| 2.2.9 Installation of SLES10 Linux MAPSRV .....                                       | 20   |
| 2.2.10 Installation of Tivoli Service Automation Manager on MAPSRV .....              | 20   |
| 2.2.11 Creating the prototype and MASTER image .....                                  | 21   |
| 2.2.12 Creating the MASTER user ID .....  | 23   |
| 2.2.13 Installing Master RPM onto the Linux Master ID .....                           | 24   |
| 2.2.14 RACF settings .....  | 24   |
| <b>Chapter 3. Configuring Managed From (Tivoli Service Automation Manager)</b> .....  | 27   |
| 3.1 Completing the initial configuration steps .....                                  | 28   |
| 3.2 Set system properties .....   | 28   |
| 3.3 Completing installation of Tivoli Service Automation Manager .....                | 30   |
| 3.4 DCM-related tasks .....   | 30   |
| 3.4.1 Preparing XML for import into Data Center Model .....                           | 31   |
| 3.4.2 Importing XML .....   | 34   |
| 3.4.3 Finalizing the Data Center Model .....  | 34   |
| 3.5 Registering an image .....  | 34   |
| <b>Chapter 4. Configuring Managed To</b> .....  | 37   |
| <b>Chapter 5. Step-by-step checklist</b> .....  | 39   |
| 5.1 Familiarize yourself with the format of the steps presented in this chapter. .... | 41   |
| 5.2 Gather the information needed to complete these steps .....                       | 41   |

|   |           |
|---|-----------|
| 5.3 Logon as MAINT . . . . .  | 42        |
| 5.4 Attach and format DASD . . . . .  | 42        |
| 5.5 Set up the SYSTEM CONFIG file . . . . .                                   | 42        |
| 5.6 Edit the SYSTEM CONFIG file – LAN and TCPIP specifications . . . . .      | 43        |
| 5.7 Edit the SYSTEM CONFIG file – FEATURES and SET specifications . . . . .   | 43        |
| 5.8 Edit the SYSTEM CONFIG file – Virtual Network Device Management . . . . . | 44        |
| 5.9 Release and Return SYSTEM CONFIG . . . . .                                | 44        |
| 5.10 Update TCP/IP settings. . . . .  | 44        |
| 5.11 Update additional TCP/IP settings . . . . .                              | 45        |
| 5.12 More TCP/IP settings . . . . .   | 45        |
| 5.13 SYSTEM DTCPARMS settings. . . . .  | 46        |
| 5.14 Prepare to configure using DIRMAINT – retrieve CONFIGxx . . . . .        | 46        |
| 5.15 Prepare to configure using DIRMAINT – receive CONFIGxx . . . . .         | 46        |
| 5.16 Prepare to configure using DIRMAINT – Edit the CONFIGxx file . . . . .   | 47        |
| 5.17 Prepare to configure using DIRMAINT – Return the CONFIGxx file. . . . .  | 47        |
| 5.18 Set up the EXTENT CONTROL file . . . . .                                 | 47        |
| 5.19 Activate the changes to the EXTENT CONTROL file . . . . .                | 48        |
| 5.20 Define MAPSRV . . . . .  | 48        |
| 5.21 Define MAPAUTH . . . . .   | 48        |
| 5.22 Add MAPSRV and MAPAUTH to the User Directory . . . . .                   | 49        |
| 5.23 Install SLES10 on MAPSRV . . . . .                                       | 49        |
| 5.24 Install Tivoli Service Automation Manager on MAPSRV . . . . .            | 49        |
| 5.25 Authorize MAPAUTH to issue the SMAPI commands . . . . .                  | 50        |
| 5.26 Create the prototype and MASTER image . . . . .                          | 50        |
| 5.27 Create the prototype – LINDFLT DIRECT . . . . .                          | 50        |
| 5.28 Create the MASTER z/VM user ID – SL10MSTR DIRECT. . . . .                | 51        |
| 5.29 Install IBM-System-z.MASTER-1.0-1.s390xr on the MASTER. . . . .          | 52        |
| 5.30 Establish appropriate RACF settings . . . . .                            | 52        |
| 5.31 RACF configuration – edit DTCPARMS for VSMERVE and DIRMAINT . . . . .    | 54        |
| 5.32 Add RACF commands CONFIGA DATADVH. . . . .                               | 54        |
| 5.33 Summary. . . . .   | 55        |
| <b>Appendix A. Appendix . . . . .</b>   | <b>57</b> |
| A.1 Example of DCM/ XML specifications. . . . .                               | 58        |
| A.2 Common configuration errors . . . . .                                     | 63        |
| A.3 Stopping and starting Tivoli Service Automation Manager. . . . .          | 67        |
| A.4 Capacity planning and z/VM overcommit . . . . .                           | 68        |
| <b>Related publications . . . . .</b>   | <b>69</b> |
| IBM Redbooks documents . . . . .  | 69        |
| Other publications . . . . .  | 69        |
| Help from IBM . . . . .   | 69        |

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM web sites are provided for convenience only and do not in any manner serve as an endorsement of those web sites. The materials at those web sites are not part of the materials for this IBM product and use of those web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

|         |   |                 |
|---------|---|-----------------|
| DB2®    | Redbooks®   | WebSphere®      |
| ECKD™   | Redpaper™   | z/Architecture® |
| IBM®    | Redbooks (logo)  ® | z/OS®           |
| Maximo® | System z®   | z/VM®           |
| RACF®   | Tivoli®   |                 |

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



# Preface

Cloud computing, using shared resources in public spaces instead of in-house IT organizations, is the latest thing in IT. Lines of business even bypass their own IT shops to take advantage of external providers of cloud offerings. However, many of the users that employ public cloud services have not considered issues involving security, compliance, and availability.

Cloud represents a new business model that requires a process discipline as well as the use of a corresponding set of technologies. The new model requires an understanding of the hardware configuration, software images, a virtualized storage infrastructure, and network management.

For many organizations that have mainframe resources, the IT professionals already manage these different disciplines and aspects of resources as part of their overall management of the platform. The mainframe's proven capability to efficiently and securely provide virtualization, combined with the existing skills in the IT organization, suggest that in-house mainframe resources provide an ideal environment in which to pilot cloud computing.

This IBM® Redpaper™ document describes the steps we took to create an environment that can efficiently deploy and manage a cloud in a Linux®-based Infrastructure as a Service (IaaS).

## The team who wrote this paper

This paper was produced by a team of specialists working at the International Technical Support Organization, Poughkeepsie Center.

**Mike Buzzetti** is an IT Architect at the IBM Design Center with worldwide focus on client enterprise infrastructures. He began his IBM career in 2003 at the Test and Integration Center for Linux. In 2006 Mike joined the Design Center, where he helps customers architect and design optimized IT Infrastructures. He designed a number of infrastructures that featured Linux on the mainframe and has had extensive experience helping clients leverage virtualization in complex environments. More recently, Mike has been a leader in implementing Cloud Computing.

Mr. Buzzetti has authored a book on J2EE on z/OS® security as well as a number of whitepapers. He is a regular presenter at user conferences and a number of IBM sponsored venues. Prior to joining IBM, Mr Buzzetti was a System Administrator and Programmer for the Chemung County (NY) Department of Social Services.

**James Kuchler** is a Systems Support Specialist currently working on Cloud Computing related projects on System z® using z/VM® and Tivoli® branded products.

Previously, James worked for 6 years as a Performance Tester for z/Linux and z/OS, where he supported the LSPR's mission of assessing relative processor capacity for several different benchmarks and System z models. In the role of performance tester, he also worked with another team to discover performance flaws in pre-release components of z/OS, while providing feedback to development and contributing to whitepapers. James holds a B.S. in Computer Science from the State University of New York at New Paltz.

**Charlie Lawrence** joined IBM in 1966 and has been involved in various aspects of systems design, development, and testing as well as in related curriculum and course development efforts that date back to the early days of System/360.

As an IBM instructor, much of Charlie's focus has been in training both new hires and experienced systems development programmers to prepare them for development and support positions in both the VM and z/OS lineage of systems. He served multiple assignments as a staff member of the IBM Kingston and IBM Mid-Hudson Valley Education Organizations, eventually based at IBM Poughkeepsie. He was also an adjunct instructor at Ulster County Community College (Stone Ridge, NY), where he taught various introductory computer courses as well as Assembler Language Programming.

His more recent endeavors include contributing to z/Architecture® ESAME (ESA Modal Extensions) systems assurance and verification, participating in the design and testing of the Common Event Adapter, and also the design and development of test cases to verify *management by exception* views of the Tivoli Enterprise Portal. Prior to joining the Cloud Computing project described in this paper, he was part of the Poughkeepsie-based IBM team that delivered the Predictive Failure Analysis (PFA) z/OS component.

Thanks to the following people for their contributions to this project:

Jay Brenneman  
Frank De Gilio  
Steve McGarril  
John Mullin  
Paul Sutura  
Sean Swehla  
**IBM Poughkeepsie**

Mike Ebbers, Alison Chandler  
**International Technical Support Organization, Poughkeepsie Center**

## Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks® publications in one of the following ways:

- ▶ Use the online **Contact us** review Redbooks form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an email to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- ▶ Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- ▶ Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- ▶ Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- ▶ Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>





# Introduction

Cloud has become the new reality for IT shops.

Even if they have not implemented cloud, many IT shops—perhaps even yours—are competing with external cloud offerings whether they realize it or not. This new reality has evolved as lines of businesses circumvent their own IT shops to take advantage of external providers of cloud offerings.

This is not only bad for IT, it is bad for the business because the public cloud offerings present exposures to the company that users generally do not see. Many of the users that utilize public cloud services have not considered the security, compliance, and availability concerns that often come with the use of these services. Clearly IT must start to provide the kind of services that lines of business seek in the public space in order to protect the business from the inherent dangers of those publicly offered services.

Cloud represents a new business model in addition to a deployment model. This new business and deployment model requires a process discipline as well as the use of a corresponding set of technology. IT must operate differently to position itself to manage cloud resources effectively. The new model requires an understanding of the hardware configuration, software images, a virtualized storage infrastructure, and network management. The understanding and integration of these aspects of the computing environment is often handled by different groups within the IT team, especially when those IT teams have a large distributed footprint. In many IT organizations with mainframe resources, the organization manages these different disciplines and aspects of resources as part of their overall management of the platform. This positions the mainframe and its organization as the best pioneers for internal cloud in the company.

The mainframe's historical capacity to efficiently and securely provide virtualization, combined with the mainframe operations staff's understanding and management of the infrastructure, contributes to the mainframe's position as the best contender to be that first cloud environment or environment of choice. Stated succinctly, System z management and execution efficiency lends itself to the cloud environment. This is why many IT shops are thinking of using their mainframe to pilot cloud. This paper describes how our team leveraged the following System z attributes which facilitate deploying a cloud:

- ▶ A highly virtualized infrastructure allowing multiple guests to run simultaneously with maximum efficiency.

- ▶ The ability to keep track of an image library and create new image instances quickly.
- ▶ A hypervisor environment that has a significant amount of monitoring capability allowing the operations team flexibility in operational integrity and usage accounting.
- ▶ The ability to perform capacity planning efficiently.
- ▶ The ability to take advantage of existing mainframe security environments in cloud instances.
- ▶ A mature systems management discipline that can be used to maximize cloud deployment.

Given all of these advantages, why aren't there more cloud deployments on the mainframe?

There are two basic reasons. First, traditionally the people responsible for cloud implementation have focused on distributed implementations, since that is what they have seen in the public cloud space. Second, many cloud implementers have an incorrect perception that mainframes lack sufficient cloud tools.

Also there is often a fair amount of myth and lore associated with the mainframe. Some people have the perception that the mainframe is more expensive than the distributed counterparts and believe that the only way to make cloud cost effective is to implement it on what they perceive to be inexpensive machines. Debunking these myths is not within the scope of this document.

This paper provides a detailed account of how we used mainframe technology for cloud tooling to configure an environment on which deploy and manage cloud computing.

## 1.1 The Managed From, Through, and To Component Models

Before discussing the details of implementing a cloud (or Infrastructure as a Service (IAAS)) on System z, it might be helpful if we first describe three perspectives of roles, resources, and requirements in the context of component models. Three component models in our implementation are discussed here and in the chapters that follow as the *Managed From*, *Managed Through*, and *Managed To* component models.

These component models are depicted in Figure 1-1. The legend in Table 1-1 provides a brief description for each numbered component in each model. These descriptions are provided as a high level introduction to concepts that are be described in further detail in the chapters that follow.

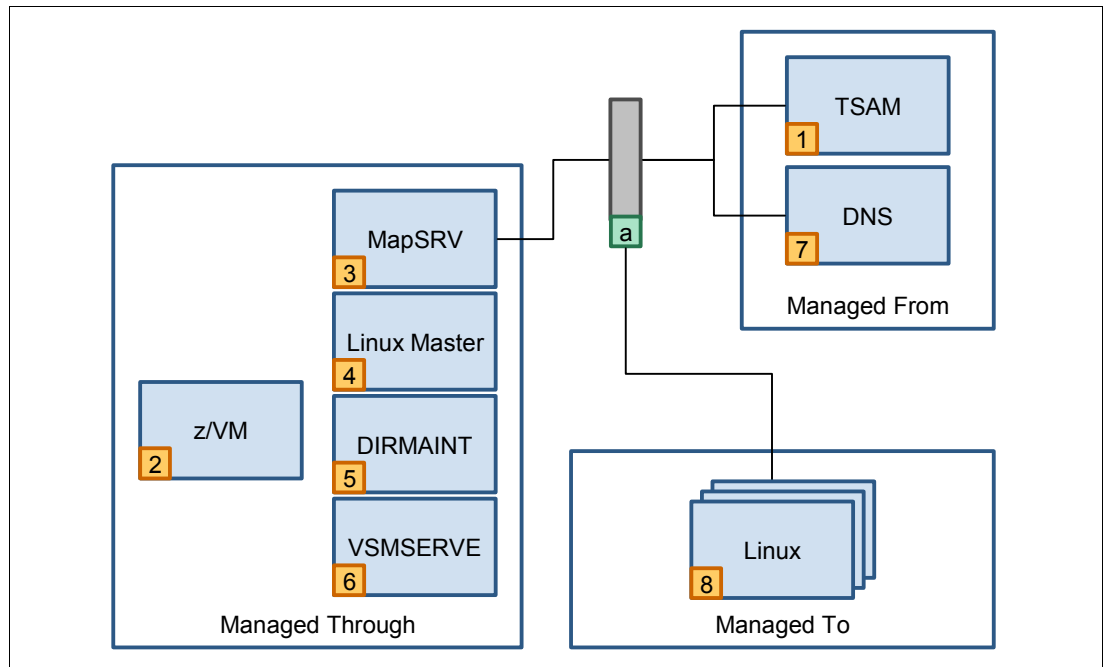


Figure 1-1 Managed From - Through - To Component Models

Table 1-1 Component Model Legend

| Component # | Description   |
|-------------|---|
| 1           | TSAM Server - This configuration item represents the crux of the cloud management. Tivoli Service Automation Manager provides the end user portal (for requesting a service such as the instantiation of a cloud of Linux instances or IAAS). It also provides the administrator portal that the cloud administrator will use to define and deploy new services |
| 2           | z/VM 5.4 - z/VM controls and manages the underlying resources for the Cloud. z/VM provides the virtual processors, memory and disks needed to create the Linux instances  |

| Component # | Description  |
|-------------|--|
| 3           | Linux instance (SLES 10 SP2) acting as the MAPSRV. MAPSRV is a Linux image that is created as part of the installation phase. Tivoli Service Automation Manager communicates to the z/VM hypervisor by way of this Linux image. During the installation phase, after SLES 10 is loaded onto this image, a special RPM is installed. The RPM provides the MAPSRV functionality. This instance can be maintained the same as any other Linux instance, but must always be available. |
| 4           | The Linux Master is a Linux instance that will be cloned into the provisioned images in the Managed To environment.  |
| 5           | DIRMAINT - this component provides z/VM user directory functions needed during the configuration of z/VM as well as during the provisioning of Linux instances.  |
| 6           | VSMSSERVE - is a component that exists on z/VM. It provides the underlying systems management Application Programming Interface (API) that MAPSRV uses to automate the provisioning of Linux instances.  |
| 7           | Domain Name Server - The DNS server will be used to store the mappings of IP address to fully qualified domain names (FQDNs). This information will be used when naming each provisioned Linux instance image.   |
| 8           | Provisioned Linux instances  |

- ▶ The *Managed From* component model shown in Figure 1-1 on page 3, refers to the collective roles, resources and requirements for Tivoli Service Automation Manager (TSAM) and Domain Name Server (DNS) items 1 and 7. Additional information for these items are provided in Chapter 3, “Configuring Managed From (Tivoli Service Automation Manager)” on page 27.
- ▶ The *Managed Through* component model shown in Figure 1-1 refers to z/VM, MAPSRV, Linux Master, DIRMAINT, and VSMSSERVE as items 2 through 6. Additional details on the *Managed Through* component model are provided in Chapter 2, “Configuring Managed Through (z/VM)” on page 7.
- ▶ The *Managed To* component model refers to the Linux instances to be provisioned (item 8). Additional details on this component are provided in Chapter 4, “Configuring Managed To” on page 37

The following sections provide an overview of each of these component models. Although integral to the overall implementation, we do not include any details relative to RACF® and networking and connectivity at this point. These topics are dealt with briefly in 2.2.14, “RACF settings” on page 24 and in 2.2.2, “Setting up the SYSTEM CONFIG file for networking” on page 11.

### 1.1.1 The Managed From component model

Although it is not shown in Figure 1-1, end users use a Web 2.0 portal to request Tivoli Service Automation Manager to provide an IAAS. Using the attributes of the request, Tivoli Service Automation Manager communicates with the *Managed Through* component to initiate



the process of dynamically creating, or *provisioning* Linux instances as needed to fulfill the requested IAAS.

After the IAAS is provisioned, that same Web 2.0 portal provides IAAS administrative functionality to the users through the Tivoli Service Automation Manager.

The DNS server is used to store the mappings of IP addresses to fully qualified domain names (FQDN). Tivoli Service Automation Manager uses this information to generate the names of each provisioned Linux instance. As an example, if a user were to make a request for a single Linux instance, Tivoli Service Automation Manager would obtain a free IP address from the pool of available addresses. The details about that pool of available addresses are discussed later in topics relating to the importing of XML and the Data Center Model. The Data Center Model serves as a repository of information about the configuration items and their associated attributes and status.

Although not shown in Figure 1-1, there are additional aspects of the Managed From component model that you should be aware of. These include the following:

- ▶ The Image Library: This library stores information about the Golden Master Images. It is composed of Data Center Model (DCM) elements such as virtual server templates, software stacks, and images.
- ▶ The Workflows: The part of Tivoli Provisioning Manager responsible for most of the automation in a standard Tivoli Service Automation Manager environment. The workflows that ship with the product automate the cloning of master images and configuring them for each service.
- ▶ The LDAP: The Tivoli Service Automation Manager uses this local LDAP repository for authentication and authorization.
- ▶ The Data Center Model: The DCM is part of Tivoli Provisioning Manager. This component stores information that is needed to provision and manage objects in the data center.

### 1.1.2 The Managed Through component model

The *Managed Through* component model includes z/VM, MAPSRV, Linux Master, DIRMAINT and VSMERVE, which essentially provide the services, or perform the utilitarian nuts and bolts functions that provision and deprovision Linux instances.

The following list provides a brief description of each of the parts of the Managed Through component model depicted in Figure 1-1 on page 3.

- ▶ z/VM controls and manages the underlying resources for the guests (MAPSRV, Linux Master, DIRMAINT and VSMERVE), providing what is often called *hypervisor* support. In our implementation this might be referred to as cloud support in the context of managing virtual processors, memory, and disks as part of the Linux instantiation.
- ▶ MAPSRV is a Linux image that is created as part of the installation phase. Tivoli Service Automation Manager communicates to the z/VM hypervisor by way of this Linux image. During the installation phase, after SLES 10 is loaded onto this image, a special RPM is installed. The RPM provides the MAPSRV functionality. This instance can be maintained in the same manner as any other Linux instance, but must always be available.
- ▶ The Linux Master is the base image that will be cloned into the provisioned images in the Managed To environment.
- ▶ VSMERVE is a component that exists on z/VM that provides the underlying Systems management Application Programming Interface (SMAPI) that MAPSRV uses to automate the provisioning of Linux instances.

### 1.1.3 The Managed To component model

The *Managed To* component model represents services that have been instantiated. In the context of this paper, this would be a set of Linux instantiations.

## 1.2 Summary

In summary, this paper shows how deployment of a cloud on System z can serve as an appropriately secure and viable source for lines of business that might otherwise seek alternative—and possibly risky—solutions outside the realm of your IT organization.

System-z-based cloud deployment can take advantage of the mainframe's historical capacity to efficiently and securely provide virtualization. When this historical perspective is combined with your mainframe operations team's understanding and management of an IT infrastructure, it can only contribute to the mainframe's position as the best contender to be your first cloud environment or environment of choice.



## Configuring Managed Through (z/VM)

This chapter describes the steps we took to prepare our Managed Through environment, specifically z/VM 5.4, which enabled us to support automated provisioning of z/VM Linux instances using Tivoli Service Automation Manager. The steps required the use of the Directory Maintenance Facility for z/VM (DIRMAINT) product.

If you are going to follow our steps, you must have DIRMAINT installed and enabled on the z/VM system that you will be using. The examples in this chapter show the steps we took to prepare a previously existing z/VM 5.4 system (called “PELV64”). You must alter some parts of the statements and commands shown here to reflect the details of your particular environment.

Chapter 5, “Step-by-step checklist” on page 39 lists the steps discussed in this chapter, but without detailed explanations. You might want to use that chapter when you are ready to proceed with the configuration process.

## 2.1 Overview of z/VM configuration

At a high level, the steps we took to configure z/VM to support virtualization for our implementation are as follows:

- ▶ Information gathering tasks:
  - Identify DASD requirements.
    - DASD for minidisks to be allocated to provisioned instances
    - DASD for two “permanent” Linux instances
  - Identify OSA requirements.
  - Identify IP address requirements.
- ▶ Configuration-related tasks:
  - Access the SYSTEM CONFIG.
  - Make changes to the SYSTEM CONFIG.
  - Make changes to EXTENT CONTROL.
  - Add VSWITCH and Virtual LAN.
  - Add definition for VSWITCH and Guest LANs.
  - Add and edit the “Features” statement in SYSTEM CONFIG.
  - Disable automatic logging off of disconnected instances.
  - Add custom user classes to SYSTEM CONFIG.
  - Apply changes to the system.
  - Update TCP/IP devices.
  - Add the IP address and routing information to the TCP/IP profile.
  - Update the AUTOLOG statement.
  - Provide PORT definitions.

## 2.2 Details of the Managed Through configuration

In preparing for the configuration of z/VM, we needed to understand and address the following requirements:

- ▶ DASD for minidisks: In our case, on PELV64, we initially had three ECKD™ mod9 volumes available. They were labeled V64M01, V64M02, and V64M03. These three volumes would be added to a “mini disk pool,” from which free space could automatically be allocated to the provisioned instances upon creation, and de-allocated and returned to the pool upon de-provisioning.
- DASD for two “permanent” Linux instances: In addition to the DASD that would support the provisioning of minidisks, we needed DASD to install two permanent Linux instances:
  - MASTER is the name of the Linux image that we used as a template for the creation of newly provisioned Linux guests.
  - MAPSRV is the name of the Linux image that serves as an intermediary between the Tivoli provisioning software and z/VM.

- ▶ OSA: Two real OSA devices were required. One would be used to login to the z/VM system. The other was attached to a z/VM VSWITCH and used for communications between MAPSRV and VSMSSERVE.
- ▶ IP addresses: We needed one IP address for each provisioned instance.

After we understood our z/VM configuration requirements, we were ready to begin reconfiguring our existing z/VM system. The remainder of this chapter provides details for that process.

## 2.2.1 Setting up the SYSTEM CONFIG file for DASD

Most of the required configuration changes were made by editing the active SYSTEM CONFIG file using the MAINT user ID. This file is accessed by CP when z/VM loads. In order to edit the file, we had to release the disk from CP, then we LINKed to it as MAINT's CF1.

When you make changes to the SYSTEM CONFIG, be sure to use the CPSYNTAX command to verify that you have no syntax errors. If you do not get a zero return code from CPSYNTAX you must correct the syntax errors before going any further. Errors in the SYSTEM CONFIG file can make your system unusable.

Example 2-1 shows the commands that release the CF1 disk from CP and allow access to it from MAINT user ID.

*Example 2-1 Gaining access to the SYSTEM CONFIG file*

---

```
CPRELEASE A
LINK * CF1 CF1 MR
ACCESS CF1 Z
```

---

Now you can edit the SYSTEM CONFIG file to enter any required modifications to define CP\_Owned and User\_Volume DASD. We used XEDIT SYSTEM CONFIG Z.

The User\_Volume\_List is a list of DASD volumes that CP should automatically attach to the system for user minidisks definitions. Because all minidisks are managed by CP, all volumes with minidisks must be attached to the z/VM system. Update the User\_Volume\_List section to list all DASD that are used for minidisks across all master, instances, and DASD pools.

We began our project with an existing z/VM system configuration with DASD as defined in Table 2-1. Note that there are three DASD (73a3 through 73A6) that were available for us to add to our configuration.

*Table 2-1 initial DASD assignments on our z/VM system*

| Address | role/usage   |
|---------|--|
| 73aa    | 201 minidisk containing the MASTER Linux image (SLES 10) |
| 73ab    | dedicated volume containing the MAPSRV Linux instance    |
| 751b    | V64M01 full pack minidisk on POOL0                       |
| 73a8    | V64M02 full pack minidisk on POOL0                       |
| 73a9    | V64M03 full pack minidisk on POOL0                       |
| 73A3    | available  |
| 73A4    | available  |
| 73A6    | available  |

Logged in as MAINT, we used the commands shown in Example 2-2 to bring our total number of DASD devices available for minidisks to six.

*Example 2-2 Commands to attach and format additional minidisks*

---

```
attach 73A3 *
attach 73A4 *
attach 73A6 *

# The cpfmtxa command will prompt you requesting formatting specifications. We
# responded PERM 0 END to the prompt from each of the following cpfmtxa commands.

cpfmtxa 73A3 V64N01
cpfmtxa 73A4 V64N02
cpfmtxa 73A6 V64N03

# After using the cpfmtxa command to format each volume, they must be detached and
# then reattached to the system.

detach 73A3
attach 73A3 system
detach 73A4
attach 73A4 system
detach 73A6
attach 73A6 system
```

---

After formatting these volumes we were ready to add them to the z/VM configuration by modifying the SYSTEM CONFIG shown in Example 2-1 on page 9.

When you modify your z/VM configuration, it is likely you will find that your SYSTEM CONFIG file is similar in that it might contain `User_volume_include` statements as shown in Example 2-3. We modified the original statement to include V64N01, V64N02, and V64N03 using the wildcard notation V64N0\*.

*Example 2-3 User\_volume\_include statement: Before and After*

---

```
User_volume_include V64M01 V64M02 V64M03 V64US1 0X73AA

# This is the “before” state. After editing SYSTEM CONFIG, the statment was modified to:

User_volume_include V64M01 V64M02 V64M03 V64US1 0X73AA V64N0*

# The V64N0* contains an “*” that serves as a wild card notation.
```

---

As a result of changing the `user_volume_include` statement, we now had the DASD assignments shown in Table 2-2. (More accurately, these are the assignments after we make the updated SYSTEM CONFIG available to CP so that our updates are recognized by the system.)

*Table 2-2 Modified DASD assignments on our z/VM system*

| Address | role/usage   |
|---------|--|
| 73aa    | 201 minidisk containing the MASTER Linux image (SLES 10) |
| 73ab    | dedicated volume containing the MAPSRV Linux instance    |
| 751b    | V64M01 full pack minidisk on POOL0                       |
| 73a8    | V64M02 full pack minidisk on POOL0                       |
| 73a9    | V64M03 full pack minidisk on POOL0                       |

| Address | role/usage                                      |
|---------|---|
| 73A3    | V64N01 full pack minidisk (soon to be in POOL0) |
| 73A4    | V64N02 full pack minidisk (soon to be in POOL0) |
| 73A6    | V64N03 full pack minidisk (soon to be in POOL0) |

Additional steps relating to these volumes (V64N01, V64N02, and V64N03) are discussed in “Setting up the EXTENT CONTROL FILE” on page 17. In that section you can see how we included these volumes as part of our “minidisk pool” known as POOL0.

## 2.2.2 Setting up the SYSTEM CONFIG file for networking

Next, we turned our attention to the statements that define the networking. This included setting up the MAPLAN LAN and granting the TCPIP and MAPSRV users permission to connect to the LAN.

Figure 2-1 provides a graphical representation of our network that you can refer to as you read about setting up the SYSTEM CONFIG file for networking.

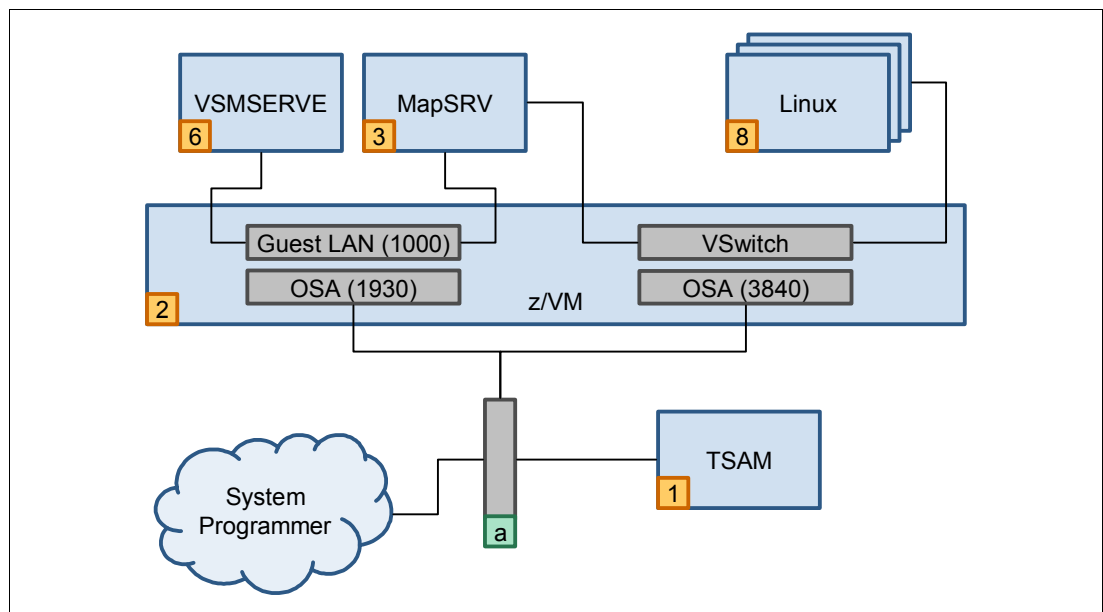


Figure 2-1 A graphic view of our networking requirements - relative to the SYSTEM CONFIG file

The MODIFY statements shown in Example 2-4 are necessary because MAPLAN needs to be defined as RESTRICTED. If you do not have similar appropriate statements in SYSTEM CONFIG, then you will need to add the statements shown in in this example to define the MAPLAN LAN. This restricts the use of MAPLAN to the z/VM user IDs of TCPIP and MAPSRV.

### Example 2-4 Define the MAPLAN LAN

```

DEFINE LAN MAPLAN OWNER SYSTEM MAXCONN 2 RESTRICTED TYPE QDIO IP
MODIFY LAN MAPLAN OWNER SYSTEM GRANT TCPIP
MODIFY LAN MAPLAN OWNER SYSTEM GRANT MAPSRV

```

Example 2-5 shows the statement that we added to the SYSTEM CONFIG file to define the VSWITCH that appears in Figure 2-1. You need to do the same, but using a name and real OSA device number appropriate for your environment.

The VSWITCH that you are defining at this point facilitates communication among the provisioned instances.

*Example 2-5 Define the VSWITCH*

---

```
DEFINE VSWITCH CLOUDSWC IP RDEV 3840
```

---

You must make additional modifications to the SYSTEM CONFIG relative to features and settings. Sample sets of commands to include or modify are provided in Examples 2-6 through 2-8.

We have included only paraphrased reasons for some of these specifications. Your best source for further details is *Tivoli Service Automation Manager V7.2 Installation and Administration Guide*, SC34-2565.

In Example 2-13 on page 15, the AUTOLOG statement identifies virtual machines that are to be started by TCPIP at the point that it begins execution. To enable this activity by TCPIP, AUTOLOG YES must be specified, as shown in Example 2-6.

*Example 2-6 Specifying Passwords\_on\_Cmds in SYSTEM CONFIG*

---

```
Passwords_on_Cmds,  
Autolog yes,  
Link yes,  
Logon yes
```

---

To support system shutdown, we also need to ensure that any virtual machine forced to disconnect will not be logged off. This required environment is established by setting the Disconnect\_timeout feature to off, as shown in Figure 2-7.

*Example 2-7 Ensure that VMs forced to disconnect will not also be logged off*

---

```
Disconnect_timeout off,
```

---

Linux instances in our environment had to shut down cleanly in the event that z/VM was shutting down. Linux instances are able to register with CP to receive a shutdown signal when z/VM is shutting down, by using the ShutdownTime and Signal ShutdownTime features shown in Example 2-8. Then z/VM waits until the time interval (in seconds) is exceeded before shutting down, or until all of the virtual machines enabled for the signal shutdown have reported a successful shutdown.

*Example 2-8 Enable Linux instances to be notified that z/VM is shutting down*

---

```
Set,  
ShutdownTime 30 ,  
Signal ShutdownTime 500
```

---



Add custom user classes to SYSTEM CONFIG to allow the MAPSRV to manage virtual network devices, as shown in Example 2-9.

*Example 2-9 Add custom user classes to SYSTEM CONFIG*

---

```

MODIFY CMD SET SUBC VSWITCH IBMCLASS B PRIVCLASS BT
MODIFY CMD QUERY SUBC * IBMCLASS B PRIVCLASS BT
MODIFY CMD IND IBMCLASS E PRIVCLASS ET
MODIFY CMD QUERY SUBC * IBMCLASS G PRIVCLASS GT
MODIFY CMD LINK IBMCLASS G PRIVCLASS GT

```

---

After modifying and saving the SYSTEM CONFIG file, the commands in Example 2-10 are used by the MAINT user ID to give the primary parm disk back to CP. When VM is re-IPLed, the settings become active.

*Example 2-10 Give the primary parm disk back to CP*

---

```

RELEASE Z
LINK * CF1 CF1 RR
CPACCESS MAINT CF1 A SR

```

---

## 2.2.3 Specifying TCP/IP devices

After z/VM is re-IPLed, you must confirm that PROFILE TCPIP contains statements to support two methods of communication: one to support 3270 login and the other for communication between provisioned instances. Figure 2-2 depicts the relationship between our OSA (DEV1930), LAN 1000, and the VSwitch. System programmers who need to login to VSMERVE and MAPSRV for administration or configuration purposes are able to do so through the OSA 1930.

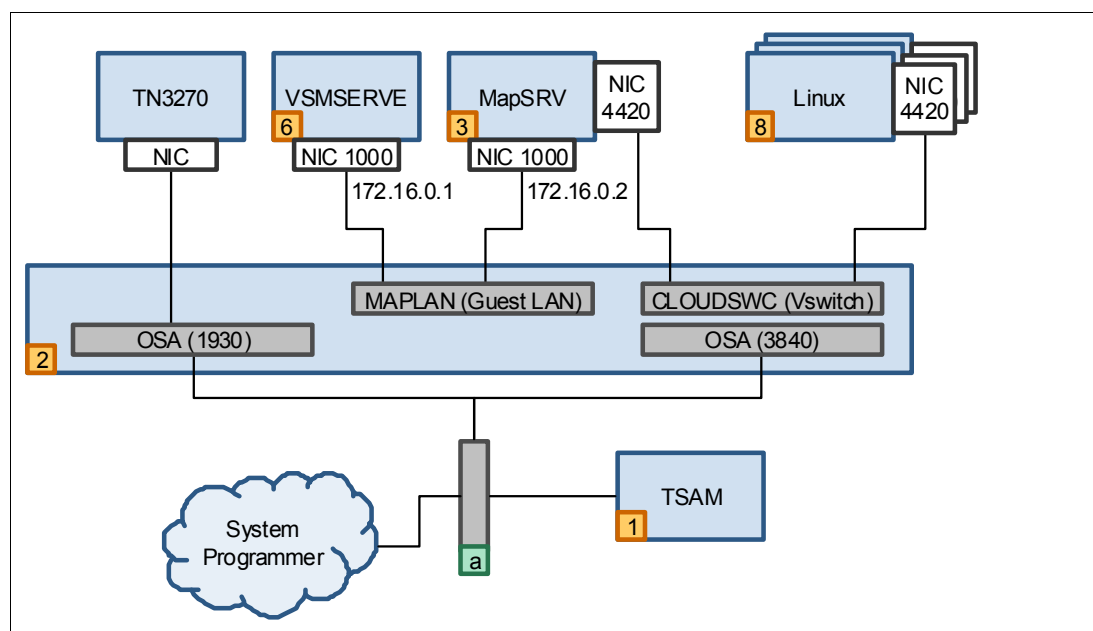


Figure 2-2 TCP/IP devices and specifications

In our case, the DEVICE and LINK statements for DEV1930 to support the configuration in Figure 2-2 were already in place, as shown in Example 2-11. This supported our ability to

login to z/VM using a 3270 emulator. It also supported communication between our provisioned Linux instances.

Likewise, we also had statements relative to MAPLAN and MAPLAND. If you do not have similar statements in your TCPIP profile, you must add them using address information for your environment. You also have to make sure that there is a START command present as shown in Example 2-11.

By default, this file is on TCPMAINT's 198 disk. In our case, the file is called PELV64 TCPIP.

*Example 2-11 Commands that need to be in the active TCPIP PROFILE*

---

```
DEVICE DEV1930 OSD 1930
LINK LNK1930 QDIOETHERNET DEV1930
DEVICE MAPLAN OSD 1000 PORTNAME LANMAP NONROUTER
LINK MAPLAND QDIOETHERNET MAPLAN MTU 1500
START DEV1930
START MAPLAN
```

---

**Note:** Device 1930 in Example 2-11 is the address of a real OSA device that has physical external connectivity for a systems programmer (for instance 3270). Your address for external communications might be different. Keep this in mind when you see references elsewhere in this paper to 1930. You might need to adjust other statements and commands to refer to your address. In our environment this line was already defined and we did not have to modify any related definitions. In the second definition, the device 1000 is not associated with a real OSA. On your system, this can be any device number that is not currently used. This number will match the address of the virtual NIC that is attached to MAPLAN by MAPSRV. We had to define the 1000 device and associated parameters.

## 2.2.4 Updating additional TCP/IP settings

Home and gateway information is specified as shown in Example 2-12. Ensure that the specifications for your environment are consistent with the statements shown in Example 2-11 and the network view you plan to establish, which is depicted for our implementation in Figure 2-2 on page 13. Do this with the following procedures:

- ▶ Modify the IP address (shown here as 129.40.178.64) and replace LNK1930 in the sample statement 129.40.178.64 255.255.255.0 LNK1930, so that they match your environment and are consistent with your version of the LINK LNK1930 QDIOETHERNET DEV1930 statement.
- ▶ The second sample statement (172.16.0.1 255.255.255.0 MAPLAND) can be used as is provided it corresponds to the XML-specified IP address for the SMAPI. Likewise, MAPLAND must correspond with a DEVICE statement that you used for the OSD 1000.

*Example 2-12 IP address and routing information in the TCP/IP PROFILE*

---

```
HOME
129.40.178.64 255.255.255.0 LNK1930
172.16.0.1 255.255.255.0 MAPLAND
GATEWAY
; Default for everything else
DEFAULTNET 129.40.178.254 LNK1930 1492 0
```

---

As previously stated, the 1930 device was already defined in our environment. We did not need to make any modifications relative to the networking addresses or device 1930.

Example 2-13 displays the AUTOLOG statement, which identifies the virtual machines to be started by the TCPIP virtual machine when it begins execution. If your TCP/IP PROFILE already has an AUTOLOG statement, make sure that FTPSERVE, PORTMAP, and VSMSSERVE are included on the AUTOLOG statement. If you do not have an AUTOLOG statement you can use this example.

Likewise, the information provided in the PORT statements must also be placed in the TCP/IP PROFILE for the specified users.

*Example 2-13 Required AUTOLOG and PORT specifications*

---

```
AUTOLOG
FTPSSERVE 0 ; FTP Server
PORTMAP 0 ; PORTMAP Server
VSMSSERVE 0 ; VM SMAPI Server
ENDAUTOLOG

PORT
20 TCP FTPSSERVE NOAUTOLOG ; FTP Server
21 TCP FTPSSERVE ; FTP Server
23 TCP INTCLIEN ; TELNET Server
111 TCP PORTMAP ; Portmap Server
111 UDP PORTMAP ; Portmap Server
172.16.0.1 845 TCP VSMSSERVE ; VM SMAPI SERVER
```

---

Note: 172.16.0.1 is the IP address that will be used to communicate with SMAPI.

That completes the specifications you will have to provide for the TCP/IP PROFILE. However, the MAPLAN NIC and the OSA still must be attached to the TCP/IP stack. This is accomplished by adding some information to SYSTEM DTCPARMS as described in the following section.

## 2.2.5 SYSTEM DTCPARMS relative to TCP/IP stack

We needed to add a reference to the MAPLAN NIC to the SYSTEM DTCPARMS. You might need to do some investigation to determine the name of the DTCPARMS file for your environment. In our case it was PELV64 DTCPARMS, where our system name was also PELV64. By default, the file resides on TCPMAINT's 198 disk.

In Example 2-14, you see the Vnic information that we added to the existing DTCPARMS file on our system; PELV64 DTCPARMS. We made no other changes. You might need to add a similar :Vnic specification to your DTCPARMS.

*Example 2-14 Vnic information*

---

```
:nick.TCPIP :type.server :class.stack
:Vnic.1000 TO SYSTEM MAPLAN
:attach.1930-1933
```

---

Note: We added the line that begins with :Vnic. The 1000 is the number that we assigned to the MAPLAN device earlier in our process and 1930 is the real OSA device address that we specified on the other DEVICE statement. (DEV1930)

## 2.2.6 Configuring DIRMAINT

As stated at the beginning of this document, the DIRMAINT product must be enabled on the z/VM system that will host the provisioned guests. DIRMAINT will be used to manage the

disks on which the provisioned instances will reside, in a way which allows manipulation via the z/VM System Management API (SMAPI) which runs on the VSMSSERVE machine. DIRMAINT also allows you to define what is essentially a template directory entry, known as a *prototype*. This prototype facilitates the cloning of virtual machines with consistent or matching attributes. In our case this made it possible for us to clone MASTER to provide the provisioned instances with known standardized attributes.

Until this point, all of the configuration was done by editing existing files. DIRMAINT maintains live versions of its configuration including the user directory. To make updates to that configuration, you must first have DIRMAINT send the configuration to your MAINT user ID's virtual reader. Then receive it into a file that you can edit (see Example 2-15). After you have made changes to your copy of the configuration, send the changes back to DIRMAINT and subsequently activate the updated configuration.

---

*Example 2-15 Retrieving the CONFIGxx DATADVH file for update*

---

```
dirm send configa datadvh

# The command must complete with RC=0. After a moment, you should see messages
# similar to the following:

DVHMT1191I Your SEND request has been sent for processing.
Ready; T=0.01/0.01 16:33:34
DVHREQ2288I Your SEND request for MAINT at * has been accepted.
RDR FILE 0139 SENT FROM DIRMAINT PUN WAS 2487 RECS 0035 CPY 001 A NOHOLD NOKEEP
DVHREQ2289I Your SEND request for MAINT at * has completed; with RC = 0.

# Receive and save the file using spoolid (139) from messages shown.

receive 139 = = a (repl
```

---

After receiving the CONFIGxx DATADVH file, edit it to conform with the requirements and examples that follow. The xx in the filename is just a suffix that is used to support the presence of multiple files of this type. They are searched in reverse alphabetical order. In our case, we only had one of these files and its name was CONFIGA DATADVH.

Two statements that need to be in CONFIGxx DATADVH are shown in Example 2-16.

---

*Example 2-16 Statements to add to CONFIG DATADVH*

---

```
ALLOW_ASUSER_NOPASS_FROM= VSMSSERVE *
ASYNCHRONOUS_UPDATE_NOTIFICATION_EXIT.UDP= DVHXNE EXEC
```

---

Additional settings that must be specified in the CONFIGxx DATADVH file are shown in Example 2-17. In these sample statements, note that DASD\_ALLOCATE= EXACT\_FF is an optional statement. If it is included, then during allocation of DASD, there will be a performance hit because the system will search for an available area on DASD that is an exact fit rather than carving out DASD space from a larger area. The benefit received in exchange for this performance hit is that the exact fit method of allocation will prevent fragmentation of DASD.

---

*Example 2-17 More statements to add to CONFIG DATADVH*

---

```
RUNMODE= OPERATIONAL
ONLINE= IMMED
DASD_ALLOCATE= EXACT_FF
DATAMOVE_MACHINE= DATAMOVE * *
MAXIMUM_UNASSIGNED_WORKUNITS= 100
```

---

After setting the CONFIGxx file to contain your specification, you are ready to send the changes back to the system. After that, you can activate the changes. This process is shown in Example 2-18.

*Example 2-18 Sending/Activating updates for CONFIGxx*

---

```
dirm file configa datadvh
    # configa datadvh is the name of your configuration file.
dirm rldcode
dirm rlddata
```

---

## 2.2.7 Setting up the EXTENT CONTROL FILE

After completing the steps relating to defining and formatting the minidisks (see Example 2-2 and Example 2-3 on page 10), we had to tell CP to use the three new minidisks as members of the already existing POOL0. This is similar to the process for updating the CONFIGxx file. Use Directory Maintenance to extract the contents of the EXTENT CONTROL and receive the file for editing.

In our implementation, we modified the extracted content to reflect our use of volumes V64N01 through V64N03. After making the changes, we provided those updates to CP using the DIRM command. The sequence of commands used are summarized in the following examples.

*Example 2-19 Extract and receive the contents of the EXTENT CONTROL file*

---

```
DIRM SEND EXTENT CONTROL
RECEIVE xxx = = A
```

---

We edited the EXTENT CONTROL file to add the 3 lines shown in Example 2-20: one for each of the new volumes we wanted to add. Then we appended 000004 000005 000006 to the end of the line that starts with POOL0, resulting in this statement:

```
POOL0 000001 000002 000003 000004 000005 000006
```

*Example 2-20 Editing the EXTENT CONTROL file*

---

```
000004 V64N01 0001 10016 ; 3390-09
000005 V64N02 0001 10016 ; 3390-09
000006 V64N03 0001 10016 ; 3390-09
```

---

Example 2-21 summarizes our resulting definitions.

*Example 2-21 Define results in EXTENT CONTROL file*

---

```
:REGIONS.
*RegionId VolSer RegStart RegEnd Dev-Type Comments<
000001 V64M01 0001 10016 3390-09
000002 V64M02 0001 10016 3390-09
000003 V64M03 0001 10016 3390-09
000004 V64N01 0001 10016 3390-09
000005 V64N02 0001 10016 3390-09
000006 V64N03 0001 10016 3390-09
:END.
:GROUPS.
*GroupName RegionList
```

---

```
POOL0 (ALLOCATE ROTATING)
POOL0 000001 000002 000003 000004 000005 000006
:END.
```

---

Example 2-22 shows the commands needed to activate the updates.

*Example 2-22 Sending/Activating updates for EXTENT CONTROL*

---

```
DIRM FILE EXTENT CONTROL
DIRM RLDEXTN
```

---

**Note:** The only other DIRMAINT-related action that we had to complete was the purging of users. Our initial success in provisioning Linux instances was coupled with unsuccessful de-provisioning. Subsequent attempts to provision would fail due to lack of resources. If you encounter a similar situation, you can recover by manually deallocating the resources associated with the Linux instances. This is accomplished using the DIRM command. Login as MAINT and issue the command **DIRM FOR provided\_userid PURGE CLEAN** for each instance (specified as a **provided\_userid**) that was provisioned. Before doing this, make sure that no one is logged in with those user IDs.

## 2.2.8 Creating MAPSRV and MAPAUTH

You need to define two z/VM guests: one called MAPSRV and the other MAPAUTH.

MAPSRV is the Linux instance by which Tivoli Service Automation Manager interfaces with z/VM. It does this using both vmcp and SMAPI. vmcp is a program that allows CP and CMS commands to be invoked directly from a Linux shell. System Management API (SMAPI) runs on VSMERVE and facilitates DIRMAINT commands to be issued from the shell. The MAPAUTH ID is used for authentication relative to SMAPI requests.

These two instances, user IDs, are defined to z/VM by two files that serve as their directory entries. Create two files, MAPSRV DIRECT and MAPAUTH DIRECT, then make any required changes to the sample files provided in Examples 2-23 and 2-24. Refer to the corresponding customization notes in Table 2-3 and Table 2-4 on page 19.

*Example 2-23 Sample file MAPSRV DIRECT file*

---

```
USER MAPSRV WELCOMES 512M 1G GT
INCLUDE IBMDFLT
IPL 73AB 1
MACHINE ESA
OPTION LNKNOPAS LANG AMENG 2
DEDICATE 73AB 73AB
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
SPECIAL 1000 QDIO 3 SYSTEM MAPLAN
AMDISK 0191 3390 AUTOV 10 V64M01 MR
AMDISK 0151 3390 AUTOV 200 V64M01 MR
AMDISK 0192 3390 AUTOV 50 V64M01 MR
```

---

- <sup>1</sup> Replace 73AB with the DASD address (this might be the address of a whole volume or of a minidisk) of the device on which you plan to install MAPSRV.
- <sup>2</sup> On the surface this might appear to be a dangerous move because it gives MAPSRV unrestricted access. However, LNKNOPAS must be used in this implementation to permit MAPSRV to LINK to any disk without a password. This facilitates MAPSRV's role to act essentially as a VM Service Machine.

*Example 2-24 Sample file MAPAUTH DIRECT file*

```
USER MAPAUTH PASSWORD 32M 32M G
INCLUDE CMSUSER
```

Table 2-3 and Table 2-4 provide customization notes to assist you establishing the MAPSRV and MAPAUTH user IDs.

*Table 2-3 Customization notes for sample file MAPSRV DIRECT*

| Command / Statement                       | Notes   |
|---|---|
| USER MAPSRV WELCOMES 512M 1G GT           |   |
| INCLUDE IBMDFLT                           |   |
| IPL 73AB                                  | Replace 73AB with the DASD address (this might be the address of a whole volume or of a minidisk) of the device on which you plan to install MAPSRV.  |
| MACHINE ESA                               |   |
| OPTION LNKNOPAS LANG AMENG                | On the surface this might appear to be a dangerous move because it gives MAPSRV unrestricted access. However, LNKNOPAS must be used in this implementation to permit MAPSRV to LINK to any disk without a password. This facilitates MAPSRV's role to act essentially as a VM Service Machine.  |
| DEDICATE 73AB 73AB                        | Replace 73AB with the DASD address (this might be the address of a whole volume or of a minidisk) of the device on which you plan to install MAPSRV.  |
| NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC | Replace 4420 and CLOUDSWC as follows:<br>CLOUDSWC should be the same value that you specified on the DEFINE VSWITCH statement as described in the section Setting up the SYSTEM CONFIG file. Likewise, make sure that the value you choose (if different from 4420) corresponds to the value you specified on the NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC statement of the LINDFLT DIRECT file (refer to discussion of Creating the prototype and MASTER image) as well as in the SL10MSTR DIRECT file. |
| SPECIAL 1000 QDIO 3 SYSTEM MAPLAN         | Replace 1000 with the value you specified in the SYSTEM DTCPARMS file (refer to discussion of Updating TCP/IP Settings  |
| AMDISK 0191 3390 AUTOV 10 V64M01 MR       | Replace with the volume label where you want MAPSRV's 191 user minidisk to reside.  |
| AMDISK 0151 3390 AUTOV 200 V64M01 MR      | Replace with the volume label of where you want MAPSRV's 151 user minidisk to reside.   |
| AMDISK 0192 3390 AUTOV 50 V64M01 MR       | Replace with the volume label of where you want MAPSRV's 192 user minidisk to reside.   |

*Table 2-4 Customization notes for sample file MAPAUTH DIRECT*

| Command / Statement             | Notes  |
|---------------------------------|--|
| USER MAPAUTH PASSW0RD 32M 32M G |  |
| INCLUDE CMSUSER                 | You might want to include IBMDFLT instead of CMSUSER, if the CMSUSER profile doesn't exist on your system. |

After creating the MAPSRV and MAPAUTH DIRECT files, use the DIRMAINT command to add them as users. This can be accomplished by entering the two commands shown in Example 2-25.

*Example 2-25 Adding MAPSRV and MAPAUTH to the directory*

---

```
dirm add mapsrv
dirm add mapauth
```

---

## 2.2.9 Installation of SLES10 Linux MAPSRV

At this point you should install SLES10 Linux on the MAPSRV ID. Note that it must be installed on the bootable address that you specified on the IPL directory statement as in Example 2-23 on page 18. The Linux installation process is beyond the scope of this document. Information relative to this subject can be found at [ibm.com](http://ibm.com); search for “Getting Started with Linux on System z.”

After SLES10 is installed on MAPSRV, verify that the vmcp command is working. This can be accomplished by entering **vmcp q time** at the command prompt. If vmcp is working properly, you should see a response similar to that shown in Example 2-26.

*Example 2-26 Using vmcp q time response to verify that vmcp is functioning*

---

```
vmcp q time
-----
TIME IS 09:56:35 EDT THURSDAY 09/23/10
CONNECT= 99:59:59 VIRTCPU= 025:48.13 TOTCPU= 027:24.68
```

---

If the vmcp command fails to return any output, add vmcp to the list of modules assigned to the MODULES\_LOADED\_ON\_BOOT variable in `/etc/sysconfig/kernel` and retry the command.

## 2.2.10 Installation of Tivoli Service Automation Manager on MAPSRV

Install Linux on MAPSRV, then copy `IBM-System-z.MAPSRV-1.0-1.s390x.rpm` (`/opt/IBM/tivoli/tpm/repository`) to the MAPSRV ID and install it using the command shown in Example 2-27.

*Example 2-27 Using rpm to install Tivoli Service Automation Manager*

---

```
rpm -ivh IBM-System-z.MAPSRV-1.0-1.s390x.rpm
```

---

Define the IP address 172.16.0.2 by editing the network configuration file `/etc/sysconfig/network/ifcfg-qeth-bus-ccw-0.0.1000`. It should contain the statements in Example 2-28.

*Example 2-28 Editing the MAPSRV network configuration file*

---

```
BOOTPROTO='static'
UNIQUE=''
STARTMODE='auto'
IPADDR='172.16.0.2'
NETMASK='255.255.255.0'
NETWORK='172.16.0.0'
BROADCAST='172.16.0.255'
_nm_name='qeth-bus-ccw-0.0.1000'
PREFIXLEN=''
```

---



MAPSRV communicates with VSMERVE using MAPAUTH's credentials; therefore, we needed to authorize MAPAUTH to issue SMAPI commands. You can provide this same authorization by logging on to your VSMERVE and then editing the VSMERVE AUTHLIST. Note that the content of this file is column sensitive. When you edit the file, simply replicate or copy an existing line to follow that same line. Then just overwrite the user ID while making no changes to any other portion of the line. The file should resemble Example 2-29 when you are done editing.

*Example 2-29 Authorize MAPAUTH to issue the SMAPI commands*

---

|               |               |
|---------------|---------------|
| DO.NOT.REMOVE | DO.NOT.REMOVE |
| MAINT         | ALL           |
| VSMERVE       | ALL           |
| MAPAUTH       | ALL           |

---

## 2.2.11 Creating the prototype and MASTER image

When you create and define the prototype and MASTER image, be careful to specify the same values in two different environments. In one environment you will be using z/VM's DIRMAINT. In the other, you will be using a set of XML statements that you will import into the Data Center Model using a process known as xmlimport.

At this point, it is worth repeating the methodology we used to compensate for any incorrect or conflicting XML specifications. At the same time we made corrections or modifications to the Data Center Model, we also incorporated those changes in our XML file. By doing so, we had a current XML file that could be xmlimported whenever we had to backtrack to a prior checkpoint.

In Example 2-30, note that name="zVM\_Prototype" value="LINUX" will provide a logical connection between the DCM and z/VM that facilitates the use of the prototype directory LINUX PROTODIR in the provisioning of a Linux instance with specific z/VM characteristics.

Likewise, name="zVM\_DiskOwnerId" value="SL10MSTR" informs the provisioning process where it can locate the MASTER copy of the bootable Linux image. This facilitates the copying of the MASTER to the target disk specified by name="zVM\_CloneDisks" value="201".

*Example 2-30 Sample XML snippet - MASTER image*

---

```
<image name="POK SLES10 SP2 Minidisk" image-type="Golden_Master"
description="Prepared for TSAM" locale="en_US" version="1.0"
boot-server="MAPSRV-bootserver" status="tested" is-device-model="SOAonRAMPimage"
software-module="POK SLES10" priority="1" >
<property component="KANAHA" name="zVM_CloneDisks" value="201" />
<property component="KANAHA" name="zVM_DiskOwnerId" value="SL10MSTR" />
<property component="KANAHA" name="zVM_Prototype" value="LINUX" />
<property component="KANAHA" name="zVM_SystemDisk" value="201" />
</image>
```

---

DIRMAINT allows you to define a prototype directory entry that contains directory statements that are common for each of the instances to be provisioned. If you need to provision instances with different configurations, you can define multiple prototypes.

We used just used one prototype directory entry as shown in Example 2-31 on page 22 and called it LINUX. This corresponds to the DCM reference previously discussed in the context of value=LINUX for name= "zVM\_Prototype".

---

*Example 2-31 Sample LINUX PROTODIR*

---

```
USER LINUX NOLOG 512M 2G G
INCLUDE LINDFLT
```

---

When you create your LINUX PROTODIR, note that the INCLUDE LINDFLT refers to a file called PROFILE LINDFLT, which contains additional attributes to fully define the user LINUX shown in Example 2-32 on page 22.

Note that in the file LINDFLT DIRECT (referred to in the previous example with contents described in the following example), the specification of NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC must correspond to what you specified in the MAPSRV and MAPAUTH DIRECT files.

After you create and file LINUX PROTODIR, use the command **dirm file LINUX PROTODIR** to provide the user definition to DIRMAINT.

---

*Example 2-32 Sample LINDFLT DIRECT*

---

```
PROFILE LINDFLT
CLASS G
STORAGE 512M
MAXSTORAGE 2047M
IPL 500
IUCV ALLOW
MACHINE ESA
OPTION QUICKDSP
CONSOLE 0009 3215 T
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK TCPMAINT 0592 0592 RR
```

---

When you have completed the creation of LINDFLT DIRECT you are ready to deploy that file to DIRMAINT. The commands and related notes to accomplish this are provided in Example 2-33.

---

*Example 2-33 Adding or Replacing LINDFLT DIRECT*

---

```
# Check for the existence of a user named LINDFLT:
dirm for lindflt get lock

# If the dirm command succeeds, the user exists and must be replaced; issue this
# command with the replace option:
dirm for lindflt replace

# Otherwise, issue this command:
dirm add lindflt
```

---

That defines the VM side of the template that will be used to provision instances. Next, you need to create the Linux part of the template by creating the MASTER instance.

## 2.2.12 Creating the MASTER user ID

You must create a directory entry for the MASTER. We used SL10MSTR as the name of our Master. Using the same process for defining other user IDs such as MAPSRV and MAPAUTH, create a file called SL10MSTR DIRECT A that contains the statements shown in Example 2-34.

Keep in mind that you might need to customize this set of sample statements. Refer to Table 2-5 on page 23 for an overview of those items that must conform to your installation.

*Example 2-34 Sample SL10MSTR DIRECT*

```
USER SL10MSTR PASSWORD 1024M 1024M G 64
INCLUDE IBMDFLT
CPU 00 NODEDICATE
CPU 01 NODEDICATE
IPL CMS
MACHINE ESA 4
OPTION QUICKDSP APPLMON
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
AMDISK 0191 3390 AUTOV 5 V64M01 MW
MDISK 0201 3390 1 10016 0X73AA MR ALL WRITE MULTI
```

*Table 2-5 Customization notes for sample file SL10MSTR DIRECT*

| Statement(s)                                      | Notes   |
|---|---|
| NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC         | Replace 4420 and CLOUDSWC as follows:<br><br>CLOUDSWC should be the same value that you specified on the DEFINE VSWITCH statement as described in the section Setting up the SYSTEM CONFIG file.<br><br>Likewise, make sure that the value you choose (if different from 4420) corresponds to the value you specified on the NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC statement of the LINDFLT DIRECT file (refer to discussion of Creating the prototype and MASTER image) file |
| AMDISK 0191 3390 AUTOV 5 V64M01 MW                | Replace 5 with the size of the desired minidisk and V64M01 with the valid for your installation   |
| MDISK 0201 3390 1 10016 0X73AA MR ALL WRITE MULTI | Our MASTER Linux has an entire mod 9 (0x73AA) available for installing the OS. Make sure that this statement corresponds to the address of your master address and its size.  |

When you have completed the creation of SL10MSTR DIRECT you are ready to deploy that file to DIRMAINT. The commands (and related notes) to accomplish this are provided in Example 2-35.

*Example 2-35 Adding or Replacing SL10MSTR DIRECT*

```
# Check for the existence of a user named sl10mstr:
dirm for sl10mstr get lock

# If the dirm command succeeds, then the user exists and must be replaced; issue this
# command with the replace option:
dirm for sl10mstr replace
```

```
# Otherwise, issue this command:  
dirm add s110mstr
```

---

### 2.2.13 Installing Master RPM onto the Linux Master ID

Install Linux on the MASTER ID, then copy IBM-System-z.MASTER-1.0-1.s390x.rpm (/opt/IBM/tivoli/tpm/repository) to the MASTER ID. Install it using the rpm command:

```
rpm -ivh IBM-System-z.MASTER-1.0-1.s390x.rpm
```

Ensure that the following packages are installed:

- ▶ perl-XML-DOM
- ▶ perl-HTML-Parser
- ▶ perl-HTML-Tagset
- ▶ perl-XML-Generator
- ▶ perl-XML-RegExp
- ▶ perl-XML-Writer
- ▶ perl-libwww-perl
- ▶ perl-HTML-Tagset

If one or more of these packages are not installed, they must be installed now, using YaST or or another appropriate installation method.

**Note:** You must set up /etc/resolv.conf on the MASTER Linux so that the provisioned guests have the correct DNS information. The provisioned instances get an unmodified copy of whatever MASTER has in its file system, so /etc/resolv.conf must list the correct nameservers for the provisioned instances.

### 2.2.14 RACF settings

The last part of preparing the managed through z/VM environment involves issuing RACF commands to allow access to and from certain parts of the system. The examples that follow demonstrate the RACF-related actions we took.

*Example 2-36 RACF - Permit access to system resources for DATAMOVE, TCPIP and FTPSERVE*

---

```
RAC PERMIT MAINT CLASS(VMRDR) ID(DATAMOVE) ACC(UPDATE)  
RAC PERMIT OPERATOR CLASS(VMRDR) ID(TCPIP) ACC(UPDATE)  
RAC PERMIT FTPSERVE CLASS(VMRDR) ID(FTPSERVE) ACC(CONTROL)
```

---

*Example 2-37 RACF - Permit VSMERVE to access the z/VM parm disk*

---

```
RAC SETROPTS GENERIC(VMMDISK)  
RAC PERMIT MAINT.CF1 ACC(ALTER) ID(VSMERVE)  
RAC PERMIT MAINT.CF2 ACC(ALTER) ID(VSMERVE)
```

---

*Example 2-38 RACF - Configuring network security*

---

```
# Define RACF resources for the VSWITCH and MAPLAN:  
  
RAC RDEFINE VMLAN SYSTEM.CLOUDSWC UACC(NONE)  
RAC RDEFINE VMLAN SYSTEM.MAPLAN UACC(NONE)
```

---

# If your system implements a VLAN, define a RACF resource for it, as well. In the  
# following command, [VLAN] should be replaced with the numerical identifier for your  
# VLAN. It must be 4 digits, so add leading zeroes if necessary.

```
RAC RDEFINE VMLAN SYSTEM.CLOUDSWC.[VLAN] UACC(NONE)
```

---

*Example 2-39 RACF - Reset all VMLAN definitions*

---

```
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) RESET(ALL)
RAC PERMIT SYSTEM.MAPLAN CLASS(VMLAN) RESET(ALL)

# Allow update access to MAINT and DTCVSW1:
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) ID(MAINT) ACCESS(UPDATE)
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) ID (DTCVSW1) ACCESS(UPDATE)

# Allow MAPSRV and TCPIP to connect to the VSWITCH, omitting the [VLAN] if your
# system does not implement one:
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) ID(MAPSRV) ACCESS(UPDATE)
RAC PERMIT SYSTEM.CLOUDSWC.[VLAN] CLASS(VMLAN) ID(MAPSRV) ACCESS(UPDATE)
```

---

*Example 2-40 RACF - Activate the VMLAN class*

---

```
RAC SETROPTS CLASSACT(VMLAN)
```

---

*Example 2-41 RACF - Configure DIRMAINT, DATAMOVE and VSMERVE*

---

```
# Give DIRMAINT and DATAMOVE RACF admin authority:
RAC ALU DIRMAINT SPECIAL
RAC ALU DATAMOVE OPERATIONS

# Allow VSMERVE to perform password validation:
RAC RDEFINE VMCMD DIAG0A0.VALIDATE UACC(NONE)
RAC PERMIT DIAG0A0.VALIDATE CLASS(VMCMD) ID(VSMERVE) ACCESS(READ)
RAC SETROPTS CLASSACT(VMCMD)

# Allow VSMERVE to connect to the RACF service machine:
RAC RDEFINE FACILITY ICHCONN UACC(NONE)
RAC PERMIT ICHCONN CLASS(FACILITY) ID(VSMERVE) ACCESS(UPDATE)
RAC SETROPTS CLASSACT(FACILITY)
```

---

*Example 2-42 Activate DIAG0A0*

---

```
# If protection for DIAG0A0 is not currently active, activate it by issuing:
RALTER VMXEVENT EVENTS1 DELMEM(DIAG0A0/NOCTL)
SETEVENT REFRESH EVENTS1

# DIAG0A0 is active by default. However, this setting can be changed in the currently
# active VMXEVENT profile by issuing:
RDEFINE VMXEVENT EVENTS1 ADDMEM(DIAG0A0/NOCTL)
```

---

Now you are ready to complete the RACF configuration relative to VSMERVE and DIRMAINT. Log in as VSMERVE and edit (or browse) VSMERVE DTCPARMS. Verify that the statements contained in that file look like those in Example 2-43. If they do not, make changes so that your DTCPARMS are consistent with what you have specified elsewhere in the context of this example.

---

*Example 2-43 RACF - final steps: Editing files for VSMSEVERE and DIRMAINT*

---

```
:Nick.VSMSEVERE :Type.server :Class.VSMAPI
                  :ESM_ENABLE.YES
                  :PARMS.-E
                  :Owner.VSMSEVERE
                  :Exit.VSMEXIT
:Nick.VSMAPI      :Type.class
                  :Name.Virtual System Management API server
                  :Command.DMSVSMAS
                  :Runtime.C
                  :Diskwarn.YES
                  :ESM_Validate.RPIVAL
                  :ESM_Racroute.RPIUCMS
```

---

Confirm that your CONFIGxx DATADVH file contains the commands shown in Example 2-44 on page 26. To do this, retrieve the CONFIG file from DIRMAINT as in Example 2-15 on page 16. After you edit the file to confirm or add commands as needed, use DIRMAINT to return the file to CP; dirm file configxx datadvh.

---

*Example 2-44 Additional CONFIGxx DATADVH requirements*

---

```
POSIX_CHANGE_NOTIFICATION_EXIT= DVHXPESM EXEC
LOGONBY_CHANGE_NOTIFICATION_EXIT= DVHXLB EXEC
USER_CHANGE_NOTIFICATION_EXIT= DVHXUN EXEC
DASD_OWNERSHIP_NOTIFICATION_EXIT= DVHXdN EXEC
PASSWORD_CHANGE_NOTIFICATION_EXIT= DVHXPn EXEC
RACF_ADDUSER_DEFAULTS= UACC(NONE)
RACF_RDEFINE_VMMDISK_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMPOSIx_POSIXOPT.QUERYDB= UACC(READ)
RACF_RDEFINE_VMPOSIx_POSIXOPT.SETIDS= UACC(NONE)
RACF_RDEFINE_SURROGAT_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMBATCH_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMRDR_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMMDISK_DEFAULTS= UACC(NONE)
AUDIT(FAILURES(READ))RACF_VMBATCH_DEFAULT_MACHINES= BATCH1 BATCH2
TREAT_RAC_RC.4= 0 | 4 | 30
```

---

When you have completed the configuration steps described in this chapter, you are ready to begin the configuration of the *Managed From* component model, described in the next chapter.



## Configuring Managed From (Tivoli Service Automation Manager)

This chapter describes tasks relating to the configuration of the Managed From component model (Tivoli Service Automation Manager), where we already had an installed Tivoli Service Automation Manager. The steps we describe here start from a completed initial installation of Tivoli Service Automation Manager 7.2.0.1; use the directions in the Tivoli Service Automation Manager Installation and Administration Guide to get to this point.

At a high level, the steps we used to configure the Managed From component are:

- ▶ Complete the initial configuration steps
- ▶ Set system properties (or more precisely, configure email communication)
- ▶ Complete the installation of the Tivoli Service Automation Manager components

The initial configuration steps basically consist of accepting and acknowledging licenses, specifying network configuration data, and creating passwords. Likewise, when you set system properties, you specify SMTP information as well as an administrator email address. This will essentially configure email communication to facilitate email notification from Tivoli Service Automation Manager to your administrator.

## 3.1 Completing the initial configuration steps

The initial configuration steps require you to use a session viewer (such as VNCVIEWER) to start a session where you accept Linux licensing and perform similar activities, including specifying your host IP address and passwords for root and virtuser. DB2®, TPM, and other facilities are initialized without your intervention.

## 3.2 Set system properties

After completing the initial configuration you can log in to the Maximo® Start Center, which provides the capability to set certain system properties. These include:

- ▶ mail.smtp.host - The host name of an SMTP mail server. This server will be used to send confirmation emails after a service request. These emails have the name of the provisioned Linux image as well as the Linux instance's root password.
- ▶ mxe.adminEmail - The email address of the administrator.

*Tivoli Service Automation Manager V7.2 Installation and Administration Guide, SC34-2565* contains a section that describes Tivoli Service Automation Manager post installation steps. We did not have to complete all of the steps described in that guide. Instead, we only set the system properties and then stopped and restarted Maximo. We recommend that you review SC34-2565 to see if there are any additional documented steps that apply to your particular environment before proceeding.

Follow these steps to set the system properties:

1. Login to the administrative user interface:  
`https://your-host-ipaddress-goes-here/maximo`  
Substitute the ip address you specified during the network configuration step for *your-host-ipaddress-goes-here*.
2. Navigate to System Properties by selecting **Go To** → **System Configuration** → **Platform Configuration** → **System Properties** (Figure 3-1 on page 29).
3. Set mail.smtp.host to the host name of the SMTP server that you will be using (Figure 3-2 on page 29).
  - a. Click **Filter**.
  - b. Enter mail.smtp.host as the value you want to use as a filter, essentially selecting mail.smtp.host as the property you want to change.
  - c. In the text box that is returned, enter the host name of the SMTP server that to be used. In our example this was dcx48b.pok.ibm.com.
4. Apply the changes specified by performing the following steps for *each* property that you changed:
  - a. Use Filter to find the system property that you want to apply.
  - b. Select the box next to the Property Name.
  - c. From the Select Action dropdown, select **Live Refresh** and click **OK**.
5. Restart the Maximo server to make the changes take effect. Refer to A.3, “Stopping and starting Tivoli Service Automation Manager” on page 67.



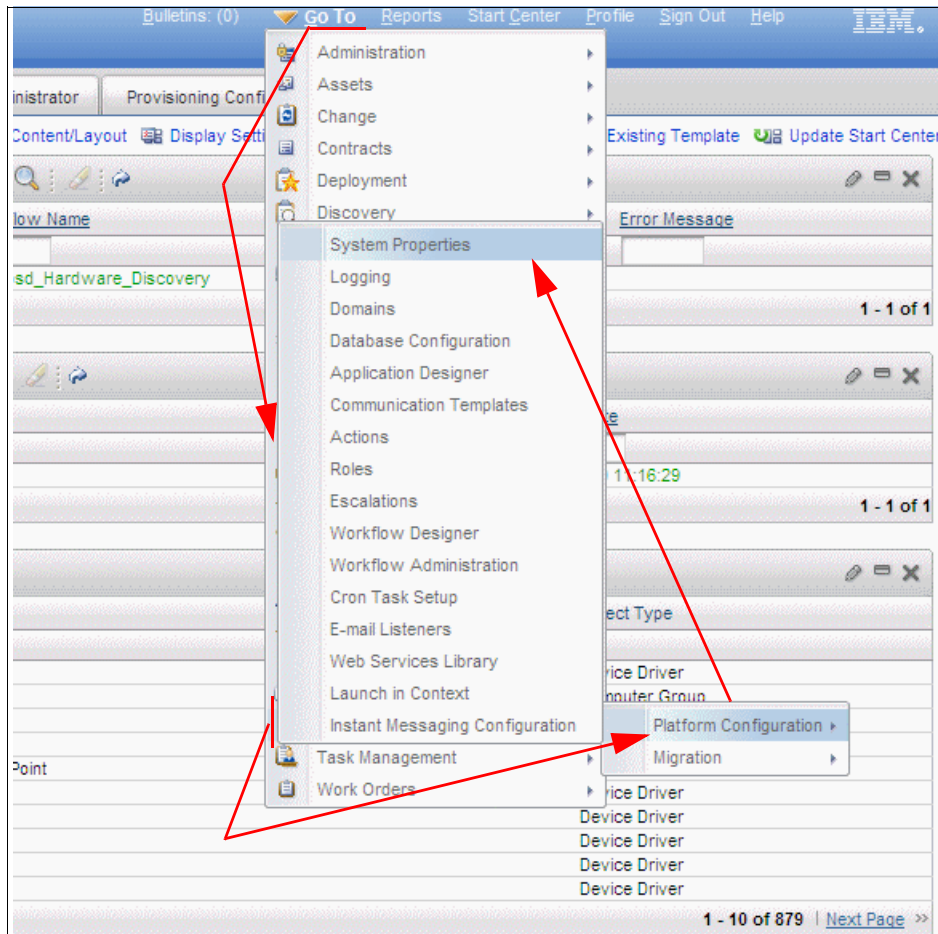


Figure 3-1 Navigating to System Properties panel

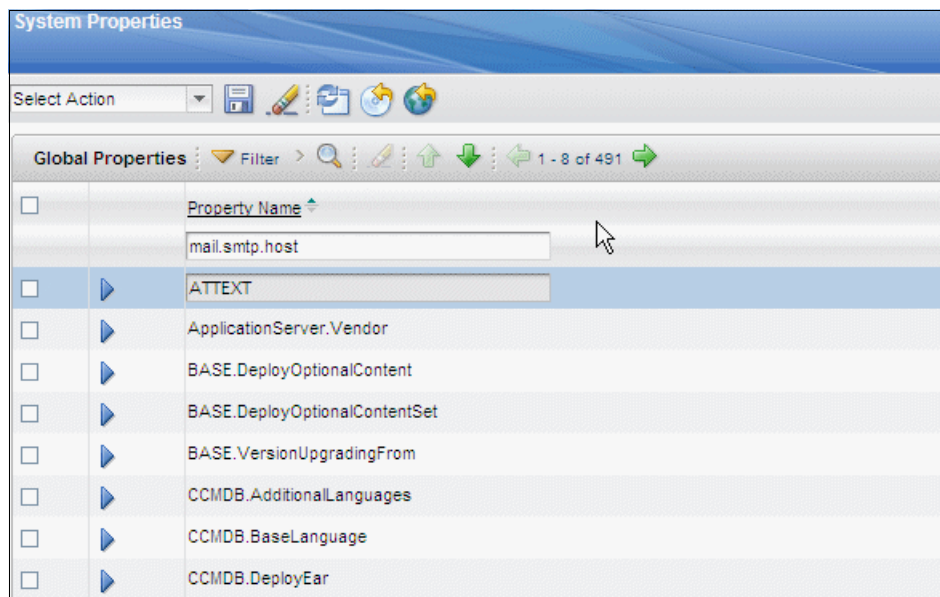


Figure 3-2 Specifying mail.smtp.host

## 3.3 Completing installation of Tivoli Service Automation Manager

This section summarizes the steps we took to complete the installation of Tivoli Service Automation Manager. We based these activities on the information provided in Chapter 2 of *Tivoli Service Automation Manager V7.2 Installation and Administration Guide*, SC34-2565. The section entitled “Completing the installation of the Tivoli Service Automation Manager Cloud Management components” describes how to configure Tivoli Service Automation Manager components that manage the virtualized environment. We recommend consulting that document if you need details about the steps described here.

Follow these steps to complete the installation:

1. Login to the WebSphere® Application Server Administration Console as wasadmin. (If Tivoli Provisioning Manager is not already running, login as tioadmin and issue this command: `/opt/IBM/tivoli/tpm/tools/tio.sh start`)
2. Navigate to the Maximo class Loading and Update Detection settings by selecting, on the left side of the panel, **Applications** → **Enterprise Applications**. On the right side of the panel, select **Maximo**.
3. Make the following selections on the panel returned:
  - Check “Reload classes when application files are updated.”
  - Enter 15 in the “Polling interval for updated files” text box.
  - Click the radio button on for “Classes loaded with parent loader first.”
  - Click the radio button off for “Classes loaded with Application class loader first.”
  - Click the radio button off for “Class loader for each WAR file in application.”
  - Set “Single class loader for application” (or confirm that it is set.)

Click **Apply**, then **Save**.

4. Login as tioadmin. Stop and start the Tivoli Provisioning Manager. The steps shown here use a shell script that is provided in the Appendix. The commands are:

```
/opt/IBM/tivoli/tpm/tools/tio.sh stop
/opt/IBM/tivoli/tpm/tools/tio.sh start
```

5. Verify that the Image Library is accessible by logging in as maxadmin to `https://your-hostname-goes-here:9443/ilrest/rest/os/TPV01IMGLIBENTRYMSTR1` and verify that XML is displayed. The XML should contain a Query tag that contains the character TPV01IMGLIBENTRYMSTR and other information.

## 3.4 DCM-related tasks

An XML file can be used to describe your resources and their relationships. After the XML is imported, the DCM represents the components used by the Managed From component model. This section describes how to modify the XML statements to reflect your specific resources, import the XML into the model using the `xmlimport` script, and the configuration steps needed to finalize the Data Center Model.

---

<sup>1</sup> Refer to the *Tivoli Service Automation Manager Installation and Administration Guide*, SC34-2565 for the exact level you are using to confirm the actual URL that you should use because it might vary from version to version.

### 3.4.1 Preparing XML for import into Data Center Model

Before you can import the XML into the Data Center Model, you must make changes or add XML statements to describe your resources. Table 3-1 provides a two column view of some of the changes you need to consider. The left column contains XML Snippets (XML out of context of the entire XML files). Note that the snippets might be incomplete and are provided in this format to point out specific keywords, in bold text, that we had to alter or specify for our implementation. Likewise, the right column provides notes about the value or relationship of the subject keyword (or keywords).

After importing the XML, all of your future manipulation of the Data Center Model will be through the Maximo interface. Do not attempt to make changes to the XML and then re-import. Instead, make necessary changes to the Data Center Model through the Maximo interface, and make corresponding changes to the XML. This will be helpful in the event that you have to start all over rebuilding the Data Center Model from scratch.

Review the XML with care and also consider the notes we provide in Table 3-1 because there are names defined in the XML that will also be referenced as part of the provisioning process. If there are discrepancies, the provisioning process will fail and you will need to edit the Data Center Model to correct the names and references.

Table 3-1 XML snippets

| XML snippet   | Usage notes/<br>relationship of bold text  |
|---|--|
| <pre>&lt;virtual-server-template&gt; &lt;resource-requirement resource-type="disk" how-many="0" size="6.9" <b>group-name="POOL0"</b> is-shared="false"&gt;</pre>  | <p>group-name="POOL0"</p> <p>The group-name refers to the name of the minidisk pool defined to z/VM. Note that the value specified for group-name must be the same as what you specified in the EXTENT CONTROL file used by DIRMAINT to define the extents of your minidisks. We used POOL0.</p>   |
| <pre>&lt;image name="POK SLES10 SP2 Minidisk" image-type="Golden_Master" description="Prepared for TSAM" locale="en_US" version="1.0" boot-server="MAPSRV-bootserver" status="tested" is-device-model="SOAonRAMPimage" software-module="POK SLES10" priority="1"&gt; &lt;property component="KANAHA" name="<b>zVM_CloneDisks</b>" value="<b>201</b>" /&gt; &lt;property component="KANAHA" name="<b>zVM_DiskOwnerId</b>" value="<b>SL10MSTR</b>" /&gt; &lt;property component="KANAHA" name="<b>zVM_Prototype</b>" value="<b>LINUX</b>" /&gt; &lt;&lt;property component="KANAHA" name="<b>zVM_SystemDisk</b>" value="201" /&gt; &lt;/image&gt;</pre> | <p>name="<b>zVM_Prototype</b>" value="<b>LINUX</b>" will provide a logical connection between the DCM and z/VM that facilitates the use of the prototype directory LINUX PROTODIR A to be used in the provisioning of a Linux instance with specific z/VM characteristics.</p> <p>Likewise, the name="<b>zVM_DiskOwnerId</b>" value="<b>SL10MSTR</b>" informs the provisioning process where it should locate the MASTER copy of the bootable Linux image, which it can then copy to the target disk specified by name="<b>zVM_CloneDisks</b>" value="201"</p> |

| XML snippet  | Usage notes/<br>relationship of bold text   |
|--|---|
| <pre> &lt;subnetwork name="zBMC Pok" ipaddress="129.40.178.0" netmask="255.255.255.0" &gt; &lt;bblocked-range from="129.40.178.1" to="129.40.178.240" /&gt; &lt;bblocked-range from="129.40.178.245" to="129.40.178.245" /&gt; &lt;bblocked-range from="129.40.178.247" to="129.40.178.247" /&gt; &lt;bblocked-range from="129.40.178.249" to="129.40.178.249" /&gt; &lt;bblocked-range from="129.40.178.250" to="129.40.178.255" /&gt; &lt;property component="KANAHA" name="gateway" value="129.40.178.254" /&gt; &lt;property component="KANAHA" name="domainname" value="pbm.ihost.com" /&gt; &lt;property component="KANAHA" name="pridns" value ="129.40.106.1"/&gt; &lt;property component="KANAHA" name="secdns" value ="129.40.106.1"/&gt; &lt;/subnetwork&gt; </pre> | <p>Each of the unblocked addresses in this subnetwork are available to be allocated to a provisioned Linux instancet</p> <p>The values <b>name="pridns" value ="129.40.106.1"</b> and <b>name="secdns" value ="129.40.106.1"</b> describe the primary and secondary domain name servers. Note that specifying these values here has no bearing on the /etc/resolv.conf file, which you must modify as a manual operation to contain these values as well.</p>   |
| <pre> &lt;resource-requirement resource-type="nic" how-many="1" size="0.0" <b>group-name="CLOUDSWC"</b> is-shared="true"&gt; &lt;property component="KANAHA" name="zVM_NetworkDeviceRange" value="3" /&gt; &lt;property component="KANAHA" <b>name="zVM_NetworkPortDeviceNumber" value="4420" /&gt;</b> &lt;property component="KANAHA" name="zVM_NetworkPortType" value="VSwitch" /&gt; &lt;/resource-requirement&gt; </pre>  | <p>These values must match or correspond to the NICDEF statement in the directory files MAPSRV DIRECT and LINDFLT DIRECT as follows:</p> <p>NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC</p> <p>The <b>group-name</b> specified in the XML must match the last token on the NICDEF statement; shown in this example as <b>CLOUDSWC</b>.</p> <p>The <b>zVM_NetworkPortDeviceNumber</b> specified in the XML must match the first token on the NICDEF statement, in this example it is shown as <b>4420</b>.</p> |

| XML snippet   | Usage notes/<br>relationship of bold text   |
|---|---|
| <pre> &lt;software-resource-template name="SLES10-S390X-SP2" software-resource- type="INSTALLATION" multiplicity- type="N" software-configuration-type="Regular" is-selected="true" is-default="false" is- deployable="true"&gt;  &lt;template-param name="zVM_Password" value="pwd012" parameter-type="String" multiplicity-type="One" is-hidden="false" is-changeable="true" is-encrypted="false"&gt;&lt;/template-param&gt;  &lt;template-param name="zLinux_RootPassword" value="pwd012" parameter- type="String" multiplicity-type="One" is-hidden="false" is-changeable="true" is-encrypted="false"&gt;&lt;/template-param&gt;  &lt;template-param name="zVM_Userid" description="zVM userid experimental" value="S10SP202" parameter-type="String" multiplicity-type="One" is-hidden="false" is-changeable="true" is- encrypted="false"&gt; &lt;/template-param&gt; </pre>                         | <p><b>name="SLES10-S390X-SP2"</b> identifies the name of the software resource. This software resource represents the operating system that is installed on the provisioned Linux instance.</p> <p><b>name="zVM_Password"</b> and <b>value="pwd012"</b> identifies the password that will be used to execute cp commands as part of the provisioning process relative to the user ID identified later as <b>name="zVM_Userid"</b> and <b>value="S10SP202"</b></p> <p><b>name="zLinux_RootPassword"</b> and <b>value="pwd012"</b> identifies the root password that will be used for a provisioned Linux instance. A new randomly generated pw is emailed to the requestor.</p> <p><b>name="zVM_Userid"</b> and <b>value="S10SP202"</b> identifies the userid that will be used on the z/VM system - and is associated with the password specified as <b>name="zVM_Password"</b> previously.</p> |
| <pre> &lt;resource name="Platform" resource-type="platform" managed="true" partitionable="false"&gt; &lt;property component="KANAHA" name="platform.architecture" value="390" /&gt; &lt;/resource&gt; &lt;resource name="CPU" resource-type="cpu" managed="true" partitionable="true"&gt; &lt;property component="KANAHA" name="cpu.family" value="s390" /&gt; &lt;property component="KANAHA" name="cpu.size" value="2" /&gt; &lt;property component="KANAHA" name="cpu.type" value="64-bit" /&gt; &lt;/resource&gt; &lt;resource name="Mem" resource-type="memory" managed="true" partitionable="true"&gt; &lt;property component="KANAHA" name="memory.size" value="4096" /&gt; &lt;/resource&gt; &lt;resource name="P00L0" resource-type="disk" group-name="vSCSI" managed="true" partitionable="true"&gt; &lt;property component="KANAHA" name="disk.size" value="35" /&gt; &lt;/resource&gt; </pre> | <p>The value specified as <b>name="P00L0"</b> (or whatever you happen to choose) needs to match the name of the minidisk pool specifications in the EXTENT CONTROL file.</p> <p>The <b>value="35"</b> describes the size in gigabytes of the volume that will be used to allocate minidisks. In our configuration (elsewhere in the XML) we specified <b>&lt;property component="KANAHA" name="MaxDiskSize" value="6.9"/&gt;</b>, which specifies the maximum size of a minidisk. This value rounds up to 7, which when divided into the 35 gives us the capacity to allocate 5 Linux instances with minidisks of size 7 on one <i>real</i> volume.</p>   |
| <pre> &lt;sap name="PING" ...&gt; &lt;default-sap operation-type="ping" /&gt; &lt;credentials search-key="master" ...&gt; &lt;password-credentials username="root" <b>password="somepassword"</b> ... /&gt; &lt;/credentials&gt; </pre>   | <p><b>password="somepassword"</b> is used to specify the password for root.</p>   |

| XML snippet   | Usage notes/<br>relationship of bold text  |
|---|--|
| <pre>&lt;sap name="SMAPI" ... &gt; &lt;credentials search-key="master" ...&gt; &lt;password-credentials username="MAPAUTH" <b>password="somepassword"</b> ... /&gt; &lt;/credentials&gt;</pre>  | <b>password="somepassword"</b> is used to specify the password for SMAPI.  |
| <pre>&lt;spare-pool <b>name="System z pool"</b>&gt; ...content omitted... &lt;/spare-pool&gt;</pre>   | <b>name="System z pool"</b> must be the same as what is specified in the vrpool.properties file; tpmPool=System z pool   |
| <pre>&lt;software-stack name="POK SLES10" locale="en_US" version="N/A" stack-type="Declared" is-device-model="Cloud Suse Linux Operating System"&gt; &lt;property component="KANAHA" name="soaonramp_vst" <b>value="5464"</b> /&gt;</pre> | At the time of the xmlimport step, you will have no idea what the ObjectID; the value to be associated with the <b>name="soaonramp_vst"</b> initially set in this example to <b>value="5464"</b> . This value needs to be changed after the xmlimport step. To accomplish this, use the Maximo DB Interface to query the ObjectID for the VST and replace the value in the software stack. |

### 3.4.2 Importing XML

Use the xmlimport.sh tool to import the XML into the Data Center Model. To use the tool, login to Tivoli Service Automation Manager server as tioadmin. Then run \$TIO\_HOME/tools/xmlimport.sh and provide the name of the XML file you want to import as the only argument.

If your import is error-free, then all of your future manipulation of the Data Center Model will be through the Maximo interface. Do not attempt to make changes to the XML and then re-import. Doing so will cause invalid duplicate entries in the Data Center Model.

Instead, make necessary changes to the Data Center Model through the Maximo interface, and make corresponding changes to the XML. This will be helpful in the event that you have to start all over at some checkpoint.

### 3.4.3 Finalizing the Data Center Model

Finalization of the Data Center Model requires the following additional configuration changes:

- The value for soaonramp\_vst (a property attribute for a component within the named POK SLES10) was originally set to 5464 during the xmlimport. It must be changed in the DCM to match the value of the OBJECTID for the Virtual Server Template (VST). This cannot be done until after xmlimport because the value is not known until the VST has been imported.
- The name specified as the spare-pool name in the XML must be the same as the name specified as the tpmPool= specification in the vrpool.properties file. Do *not* change the Data Center Model; instead, change the vrpool.properties file to match the Data Center Model.

## 3.5 Registering an image

In the Data Center Model we defined an image using the XML shown in Example 3-1 on page 35. This image represents a golden master and links the Data Center Model to the actual source disk of the Linux Master Image.

### Example 3-1 Image definition

```
<image name="POK SLES10 SP2 Minidisk" image-type="Golden_Master" description="Prepared for
TSAM" locale="en_US" version="1.0"
boot-server="MAPSRV-bootserver" status="tested" is-device-model="SOAonRAMPimage"
software-module="POK SLES10" priority="1">
<property component="KANAHA" name="zVM_CloneDisks" value="201" />
<property component="KANAHA" name="zVM_DiskOwnerId" value="SL10MSTR" />
<property component="KANAHA" name="zVM_Prototype" value="LINUX" />
<<property component="KANAHA" name="zVM_SystemDisk" value="201" />
</image>
```

For this image to be utilized by the end user as a service, a corresponding image must be registered through the Web 2.0 interface. You can do this by navigating through the following selections: **Request a New Service** → **Virtual Server Management** → **Manage Image Library** → **Register z/VM image** → **Register a new z/VM server image in the image library**.

After you select **Register a new z/VM server image in the image library**, a Register Image Input Form is displayed. Use this panel to describe the server image you want to register. A resource pool drop-down list shows all the spare pools defined in the XML. In our case, the only resource pool defined was the Poughkeepsie System z pool. Tivoli Service Automation Manager discovered the DCM image from this pool, as defined in our imported XML shown in Example 3-1. When you go through this process, you should see your image name, which might be truncated to fit into the drop-down list. If you do not see your image name, click the drop-down arrow to see all of the discovered images, which should include yours.

Other input boxes on the Register Image Input Form allow you to define the minimum and recommended values for this service. This can be specified in terms such as the amount of gigabytes of memory, number of physical CPUs, and amount of Disk Space.

When you submit the form (by clicking **OK**), the information is stored in the image defined in the Data Center Model. These values are used as the lower bounds when a user requests the service. During a service request, a user can modify the amount of CPU, memory, and disk.

After the image registration is resolved, this service is ready for use.

**Note:** z/VM provides the ability to overcommit resources. Tivoli Service Automation Manager does not expose this functionality in a straightforward manner. Refer to A.4, “Capacity planning and z/VM overcommit” on page 68.







## Configuring Managed To

The *Managed To* component model represents the provisioned Linux instances. The setup of this component model was mostly covered in Chapter 2, “Configuring Managed Through (z/VM)” on page 7. The attributes of the Managed To component result from the definitions you provided in the Managed Through and Data Center Model.

There are a few things worth pointing out. During the provisioning process, `/etc/resolv.conf` is not updated according to the values in the Data Center Model. Therefore, in order for the Managed To instances to have proper networking, the golden master must have the appropriate values in this file. The golden master is normally not running and thus does not need proper networking. The values in `/etc/resolv.conf` should be appropriate for the provisioned Linux instances.

In addition, any other configuration that is standard across all instances of this service should be built into the golden master cloned as part of the provisioning process. For instance, if the Linux instances that are provisioned as part of a service request require any specific packages (RPMs, tar files, and so forth), then these should be included on the golden master.





## Step-by-step checklist

This chapter contains a step-by-step list—including commands and statements—that can be used as a detailed guide for your installation. The list is provided with minimal descriptions and diagrams, based on the assumption that you have already read the preceding chapters.

Before starting your installation, we suggest that you review this chapter from start to finish. Pay particular attention to step 2 because this step provides a summary of items that you will need to have handy, such as device addresses, user IDs, and coffee.

The following list is a high level view of the steps used to configure your environment. The section numbering in this chapter corresponds to the step numbers in this list.

1. Familiarize yourself with the format of the steps listed in this chapter.
2. Gather the information you will need to complete these steps.
3. Logon as MAINT.
4. Attach and Format DASD.
5. Set up the SYSTEM CONFIG file.
6. EDIT the SYSTEM CONFIG file – LAN and TCPIP specifications.
7. EDIT the SYSTEM CONFIG file – FEATURES and SET specifications.
8. EDIT the SYSTEM CONFIG file – Virtual Network Device Management.
9. Release and Return SYSTEM CONFIG.
10. Update TCP/IP devices.
11. Update additional TCP/IP settings.
12. More TCP/IP Settings.
13. SYSTEM DTCPARMS settings.
14. Prepare to configure using DIRMAINT – retrieve CONFIGxx.
15. Prepare to configure using DIRMAINT – receive CONFIGxx.
16. Prepare to configure using DIRMAINT – Edit the CONFIGxx file.
17. Prepare to configure using DIRMAINT – Return the CONFIGxx file.

18. Set up the EXTENT CONTROL file.
19. Activate the changes to the EXTENT CONTROL file.
20. Define MAPSRV.
21. Define MAPAUTH.
22. Add MAPSRV and MAPAUTH to the User Directory.
23. Install SLES10 on MAPSRV.
24. Install Tivoli Service Automation Manager on MAPSRV.
25. Authorize MAPAUTH to issue the SMAPI commands.
26. Create the prototype and MASTER image.
27. Create the prototype – LINDFLT DIRECT.
28. Create the MASTER – SL10MSTR DIRECT.
29. Install IBM-System-z.MASTER-1.0-1.s390x on the MASTER.
30. Establish Appropriate RACF Settings.
31. Perform RACF configuration – edit DTCPARMS for VSMERVE and DIRMAINT.
32. Add RACF commands CONFIGA DATADVH.

## 5.1 Familiarize yourself with the format of the steps presented in this chapter

Most of the steps that follow contain examples that display commands or statements that you will issue or place into files or profiles. Notes and customization advice are interspersed with the code samples when appropriate. These notes are commented out with leading # signs and appear in a different text, as shown in Example 5-1.

*Example 5-1 Commands or statements that you will be issuing*

---

```
THISISACOMMAND nnnn SYSTEM
THISISANOTHERCOMMAND yyyy 0 end 0xnnnn

# This is a comment to help you customize these commands.

THISISANOTHERCOMMAND zzz
```

---

## 5.2 Gather the information needed to complete these steps

As you follow these step-by-step instructions, you will see notations indicating that you need to customize the commands, statements, or files with values specific to your environment or implementation. The following list summarizes the information you should have on hand before proceeding further:

- ▶ Real and Virtual minidisk addresses  
You need these so that you can ATTACH and DEFINE each minidisk you are adding to your configuration.
- ▶ The address of the disk that contains the SYSTEM CONFIG file  
Such as CF1.
- ▶ The address of your VSWITCH  
The Real Device Address of the VSWITCH.
- ▶ The VOLIDs and extents of the minidisks for POOL0  
You need to specify the VOLIDs and extents when you are editing the EXTENT CONTROL file. This same VOLID information will also be specified in the MAPSRV and MAPAUTH DIRECT files.
- ▶ The z/VM MAINT password  
The password is needed because you must LOGON as MAINT to perform configuration steps.
- ▶ The address of the real OSA  
You need to modify the TCP/IP PROFILE so that it specifies the address of the OSA that you will be using for external communications.
- ▶ The filename of the active TCP/IP PROFILE  
You need to modify the TCP/IP PROFILE to provide information relative to external communications (via the OSA) as well the device number that you have associated with the VIRTUAL NIC that is attached to MAPLAN by MAPSRV.
- ▶ The IP address for MAPLAN  
You need to specify the MAPLAN IP address in the TCP/IP stack.

- ▶ The name of the DTCPARMS file  
Because you have to modify the DTCPARMS in step 13, you will need the name of the DTCPARMS file. It is likely that the name is SYSTEM DTCPARMS.
- ▶ The VOLIDS of the volumes to be used for minidisks  
You need to specify the VOLIDS (as used in step 4 and step 18).
- ▶ The filemodes that are not currently being used when you are logged on as MAINT  
You need to specify a filemode to be used for each VOLID that you want to format when you are attaching and formatting DASD (step 4) and when setting up the EXTENT CONTROL file (step 18).

## 5.3 Logon as MAINT

To perform the steps associated with z/VM configuration, you must logon as MAINT.

## 5.4 Attach and format DASD

While logged in as MAINT, issue ATTACH and CPFMTXA commands as shown in Example 5-2. Repeat this sequence of commands for each new mini disk you are adding to your configuration. If you are not adding new mini disks, you can skip this step.

Replace *nnnn* with the real (physical) address of the volume and *vidvid* with the desired volume ID. For each volume that you attach and format with **cpfmtxa**, you will need to detach from the MAINT user ID and then attach it to the system.

These VOLIDs will be referenced again when you set up the EXTENT CONTROL file as shown in Example 5-16 on page 47.

*Example 5-2 Attaching and formatting DASD*

---

```
attach nnnn *
    # The cpfmtxa command will prompt you requesting formatting specifications. We
    # responded PERM 0 END to the prompt from each of the following cpfmtxa commands.
cpfmtxa nnnn vidvid

detach nnnn
attach nnnn system
```

---

## 5.5 Set up the SYSTEM CONFIG file

Using the MAINT user ID, issue the commands shown in Example 5-3 to release the parm disk from CP. The ACCESS command allows you to edit the SYSTEM CONFIG file as shown in the example.

*Example 5-3 Release the parm disk from CP*

---

```
CPRELEASE A
LINK * CF1 CF1 MR
ACCESS CF1 Z
XEDIT SYSTEM CONFIG Z
```

---

## 5.6 Edit the SYSTEM CONFIG file – LAN and TCPIP specifications

Using XEDIT, add the commands shown in Example 5-4 to the SYSTEM CONFIG file. These commands define the relationship of MAPLAN LAN, TCPIP, and MAPSRV users. They also define the required VSWITCH and the definitions for two more LANs.

In Example 5-4, replace 3840 in the DEFINE VSWITCH statement with your Real Device Address. Note also that the MAXCONN for MAPLAN should be set to 2. For security purposes, this restricts the number of connections for MAPLAN to only the two that it requires.

*Example 5-4 SYSTEM CONFIG LAN and TCP/IP Specifications*

---

```
DEFINE LAN MAPLAN OWNER SYSTEM MAXCONN 2 RESTRICTED TYPE QDIO IP
MODIFY LAN MAPLAN OWNER SYSTEM GRANT TCPIP
MODIFY LAN MAPLAN OWNER SYSTEM GRANT MAPSRV
DEFINE VSWITCH CLOUDSWC IP RDEV 3840
DEFINE LAN QDIOLAN OWNER SYSTEM TYPE QDIO MAXCONN 50 IP
DEFINE LAN CLOUDLAN OWNER SYSTEM TYPE HIPER MAXCONN 50 IP
```

---

## 5.7 Edit the SYSTEM CONFIG file – FEATURES and SET specifications

Using XEDIT, add or modify the Features and Set statements in SYSTEM CONFIG so that the statements shown in Example 5-5 are included in your configuration file. There might be other statements in your existing file, but those shown here *must* be in the file. Likewise, any conflicting statements should be removed.

*Example 5-5 Features and Set Specifications*

---

```
Disconnect_timeout off,

Passwords_on_Cmds,
Autolog yes,
Link yes,
Logon yes

Set,
ShutdownTime 30,
Signal ShutdownTime 500
```

---

## 5.8 Edit the SYSTEM CONFIG file – Virtual Network Device Management

Using XEDIT, add custom user classes to SYSTEM CONFIG to allow MAPSRV to manage virtual network devices as shown in Example 5-6.

*Example 5-6 Custom User Classes for Virtual Network Device Management*

---

```
MODIFY CMD SET SUBC VSWITCH IBMCLASS B PRIVCLASS BT
MODIFY CMD QUERY SUBC * IBMCLASS B PRIVCLASS BT
MODIFY CMD IND IBMCLASS E PRIVCLASS ET
MODIFY CMD QUERY SUBC * IBMCLASS G PRIVCLASS GT
MODIFY CMD LINK IBMCLASS G PRIVCLASS GT
```

---

## 5.9 Release and Return SYSTEM CONFIG

Issue the commands shown in Example 5-7 to release the parm disk from MAINT and return it to CP.

*Example 5-7 Release and Return SYSTEM CONFIG*

---

```
RELEASE Z
LINK * CF1 CF1 RR
CPACCESS MAINT CF1 A SR
```

---

## 5.10 Update TCP/IP settings

To define your two network devices (one for communication between MAPSRV and VSMSEVER and the other for external communications) you must modify your active TCPIP PROFILE. The file type is PROFILE. Use XEDIT to add the statements shown Example 5-8.

Replace 1930 with the address of a real OSA device that has physical external connectivity (to your intranet, for example). Replace 1000 with any device number that is not currently being used. This number must match the address of the virtual NIC that is attached to MAPLAN by MAPSRV.

Place the two START commands at the end of the TCPIP profile.

*Example 5-8 Updating TCP/IP Settings*

---

```
DEVICE DEV1930 OSD 1930
LINK LNK1930 QDIOETHERNET DEV1930
DEVICE MAPLAN OSD 1000 PORTNAME LANMAP NONROUTER
LINK MAPLAND QDIOETHERNET MAPLAN MTU 1500

START DEV1930
START MAPLAN
```

---



## 5.11 Update additional TCP/IP settings

Add the IP address and routing statement for the TCP/IP stack to the profile using the sample provided in Example 5-9. Modify the example using the notes provided in Table 5-1.

Table 5-1 Guide to replacing values in the example commands

| Old value                                | Replace with   |
|--|--|
| 129.40.178.64 255.255.255.0 LNK1930      | Replace with a statement that contains the IP address associated with your LNKnnnn where LNKnnnn should correspond to the LINK LNK1930 QDIOETHERNET DEV1930 as you provided it.                            |
| 172.16.0.1. 255.255.255.0 MAPLAND        | Replace 172.16.0.1 with the IP address that is to be associated with MAPLAN.<br>Replace with a statement that replaces the string MAPLAND with whatever you used in the DEVICE statement for the OSD 1000. |
| DEFAULTNET 129.40.178.254 LNK1930 1492 0 | This should correspond to 129.40.178.64 255.255.255.0 LNK1930 mentioned previously.  |

Example 5-9 TCP/IP Stack - sample content

```
HOME
129.40.178.64 255.255.255.0 LNK1930
172.16.0.1. 255.255.255.0 MAPLAND
GATEWAY
; Default for everything else
DEFAULTNET 129.40.178.254 LNK1930 1492 0
```

## 5.12 More TCP/IP settings

Add these three servers to the AUTOLOG and PORT specifications in your TCP/IP profile. Note also that the IP address 172.16.0.1 must be the same address that you specified for MAPLAND in the previous step.

Example 5-10 Required AUTOLOG and PORT specifications

```
AUTOLOG
FTPSERVE 0 ; FTP Server
PORTMAP 0 ; PORTMAP Server
VSMSERVE 0 ; VM SAPI Server
ENDAUTOLOG

PORT
20 TCP FTPSERVE NOAUTOLOG ; FTP Server
21 TCP FTPSERVE ; FTP Server
23 TCP INTCLIEN ; TELNET Server
111 TCP PORTMAP ; Portmap Server
111 UDP PORTMAP ; Portmap Server
172.16.0.1 845 TCP VSMSERVE ; VM SAPI SERVER
```

## 5.13 SYSTEM DTCPARMS settings

The SYSTEM DTCPARMS file (or whatever other *filename* DTCPARMS you are using) must be updated such that the value 1000 in the example is the same value that you assigned to the MAPLAN device. Likewise the reference to 1930-1933 should correspond to the range of values starting with the address of your real OSA device.

*Example 5-11 SYSTEM DTCPARMS - sample content*

---

```
:nick.TCPIP :type.server :class.stack  
:Vnic.1000 TO SYSTEM MAPLAN  
:attach.1930-1933
```

---

## 5.14 Prepare to configure using DIRMAINT – retrieve CONFIGxx

Until this step you have been editing existing files available to the MAINT user ID. Now you need to make changes to what is considered the live version of DIRMAINT's configuration. Have DIRMAINT send the configuration to the virtual reader of your MAINT user ID and receive it into a file that you can edit. After you make changes to your copy of the configuration, send the altered file back to DIRMAINT and then activate the updated configuration.

Start this process by issuing **dirm send** as shown in Example 5-12. Sample responses from the system are also provided here.

*Example 5-12 SYSTEM DTCPARMS - sample content*

---

```
dirm send configa datadvh  
  
# You should see messages similar to those that follow.  
  
DVHXMT1191I Your SEND request has been sent for processing.  
Ready; T=0.01/0.01 16:33:34  
DVHREQ2288I Your SEND request for MAINT at * has been accepted.  
RDR FILE 0139 SENT FROM DIRMAINT PUN WAS 2487 RECS 0035 CPY 001 A NOHOLD NOKEEP  
DVHREQ2289I Your SEND request for MAINT at * has completed; with RC = 0.
```

---

## 5.15 Prepare to configure using DIRMAINT – receive CONFIGxx

Receive the file that was placed in your reader in step 14 and save it to your A disk. Replace the file of the same name if it exists. In Example 5-13, 1309 is the spoolid of the file that was sent by DIRMAINT to MAINT's READER. You must replace 1309 with the actual spoolid of the file that you want to receive.

*Example 5-13 Receive CONFIGxx*

---

```
receive 1309 = = a (repl
```

---

## 5.16 Prepare to configure using DIRMAINT – Edit the CONFIGxx file

Edit the file that you received. Add the first two commands shown in Example 5-14 and confirm that the other commands are also in the file. Add any that are missing.

*Example 5-14 Edit CONFIGxx*

---

```
ALLOW_ASUSER_NOPASS_FROM= VMSERVE *
ASYNCHRONOUS_UPDATE_NOTIFICATION_EXIT.UDP= DVHXNE EXEC
RUNMODE= OPERATIONAL
ONLINE= IMMED
DASD_ALLOCATE= EXACT_FF
DATAMOVE_MACHINE= DATAMOVE * *
DVHDXD_FLASHCOPY_BEHAVIOR= 2
DVHDXD_FLASHCOPY_COMPLETION_WAIT= 0 0
MAXIMUM_UNASSIGNED_WORKUNITS= 100
```

---

## 5.17 Prepare to configure using DIRMAINT – Return the CONFIGxx file

After you have made the necessary changes to the CONFIGxx file, send it back to DIRMAINT and activate the changes using the commands shown in Example 5-15.

*Example 5-15 Return CONFIGxx*

---

```
dirm file configa datadvh
dirm rldcode
dirm rlddata
```

---

## 5.18 Set up the EXTENT CONTROL file

Earlier you defined and formatted new mini disks. Now you are ready to modify the contents of EXTENT CONTROL to define these volumes as members of POOL0.

When examining the following example of commands and statements, keep in mind that V64N01, V64N02, and V64N03 are the volids we added to our configuration. There were already three VOLIDS in the EXTENT CONTROL file that were known as V64M01, V64M02, and V64M03. We specified our three new volumes by inserting the three Extent Control statements for 000004, 000005, and 000006 immediately following the existing entry for 000003.

*Example 5-16 Edit EXTENT CONTROL*

---

```
DIRM SEND EXTENT CONTROL
RECEIVE xxx = = A
```

# Then edit EXTENT CONTROL to add lines similar to these:

```
000004 V64N01 0001 10016 ; 3390-09
000005 V64N02 0001 10016 ; 3390-09
000006 V64N03 0001 10016 ; 3390-09
```

```
# One additional change needs to be made to the EXTENT CONTROL file. We modified
# an existing POOL0 definition that was originally:

POOL0 000001 000002 000003

# We appended 000004 000005 000006, resulting in the following:

POOL0 000001 000002 000003 000004 000005 000006
```

---

## 5.19 Activate the changes to the EXTENT CONTROL file

The changes made to EXTENT CONTROL in the previous step are made active using the commands in Example 5-17.

*Example 5-17 Make EXTENT CONTROL changes active*

```
DIRM FILE EXTENT CONTROL A
DIRM RLDEXTN
```

---

## 5.20 Define MAPSRV

Define a z/VM guest called MAPSRV. This is the Linux instance from which Tivoli Service Automation Manager interfaces with z/VM. Define this user ID to z/VM by creating a file that serves as its directory entry.

Example 5-18 can serve as a model for your MAPSRV DIRECT file. Be sure to replace, as appropriate, such values as 73AB, 4420, and 1000, as well as disk addresses and volids to match your configuration.

*Example 5-18 Defining MAPSRV*

```
USER MAPSRV WELCOMES 512M 1G GT
INCLUDE IBMDFLT
IPL 73AB
MACHINE ESA
OPTION LNKNOPAS LANG AMENG
DEDICATE 73AB 73AB
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
SPECIAL 1000 QDIO 3 SYSTEM MAPLAN
AMDISK 0191 3390 AUTOV 10 V64M01 MR
AMDISK 0151 3390 AUTOV 200 V64M01 MR
AMDISK 0192 3390 AUTOV 50 V64M01 MR
```

---

## 5.21 Define MAPAUTH

Define a z/VM user ID called MAPAUTH, which is used for authentication. Define this user ID to z/VM by creating a file that serves as its directory entry. Use the content of Example 5-19 on page 49 to create or edit the MAPAUTH DIRECT A file.

---

*Example 5-19 Defining MAPAUTH*

---

```
USER MAPAUTH PASSWORD 32M 32M G
INCLUDE CMSUSER
```

---

## 5.22 Add MAPSRV and MAPAUTH to the User Directory

When you are done editing the MAPSRV DIRECT and MAPAUTH DIRECT files, you can add these users to the system using the commands shown in Example 5-20.

---

*Example 5-20 Adding MAPSRV and MAPAUTH to the User Directory*

---

```
dirm add mapsrv
dirm add mapauth
```

---

## 5.23 Install SLES10 on MAPSRV

Install SLES10 Linux on MAPSRV. Note that it must be bootable from the address you specified on the IPL directory statement.

The details of the Linux installation process are beyond the scope of this document. After you have completed the installation, verify that vmcp is available using the **vmcp** command as shown in Example 5-21.

---

*Example 5-21 Verifying vmcp availability after Installing SLES10*

---

```
vmcp q time
```

# In response to this command, you should see output similar to:

```
TIME IS 09:56:35 EDT THURSDAY 09/23/10
CONNECT= 99:59:59 VIRTCPU= 025:48.13 TOTCPU= 027:24.68
```

# If the vmcp command fails to return any output, add vmcp to the list of modules  
# assigned to the MODULES\_LOADED\_ON\_BOOT variable in /etc/sysconfig/kernel.

---

## 5.24 Install Tivoli Service Automation Manager on MAPSRV

Copy the IBM-System-z.MAPSRV-1.0-1.s390x.rpm file from the Tivoli Service Automation Manager management system to the MAPSRV guest and install it using the rpm command shown in Example 5-22.

---

*Example 5-22 Using rpm to install Tivoli Service Automation Manager on MAPSRV*

---

```
rpm -ivh IBM-System-z.MAPSRV-1.0-1.s390x.rpm
```

---

## 5.25 Authorize MAPAUTH to issue the SMAPI commands

Logon to VSMSSERVE, IPL CMS, and XEDIT VSMSSERVE AUTHLIST to add MAPAUTH to the VSMSSERVE AUTHLIST using the sequence of statements described in Example 5-23. *Note that the content of this file is column sensitive.* When you edit the file, simply replicate or copy an existing line to follow that same line, then overwrite the user ID while making no changes to any other portion of the line.

*Example 5-23 Authorize MAPAUTH to use SMAPI*

---

|               |               |
|---------------|---------------|
| DO.NOT.REMOVE | DO.NOT.REMOVE |
| MAINT         | ALL           |
| VSMSSERVE     | ALL           |
| MAPAUTH       | ALL           |

---

## 5.26 Create the prototype and MASTER image

Create a prototype entry (as in Example 5-24) called LINUX. This should be referenced in the DCM by setting the value for name= zVM\_Prototype to LINUX.

For details about the zVM\_Prototype, the XML, and the Data Center Model in general, refer to the pertinent sections of Chapter 3.

*Example 5-24 Creating the prototype and MASTER image*

---

```
# Create a file called LINUX PROTODIR A, and add the following:
USER LINUX NOLOG 512M 2G G
INCLUDE LINDFLT
# "File" this prototype with DIRMAINT:
dirm file LINUX PROTODIR
```

---

## 5.27 Create the prototype – LINDFLT DIRECT

Create a file called LINDFLT DIRECT A with content equivalent to Example 5-25. Note that **NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC** must correspond to what you specified in MAPSRV and MAPAUTH DIRECT files.

*Example 5-25 Creating the LINDFLT DIRECT file*

---

```
PROFILE LINDFLT
CLASS G
STORAGE 512M
MAXSTORAGE 2047M
IPL 500
IUCV ALLOW
MACHINE ESA
OPTION QUICKDSP
CONSOLE 0009 3215 T
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
```

---

```
SPOOL 000C 2540 READER *
SPOOL 000D 2540 PUNCH A
SPOOL 000E 1403 A
LINK MAINT 0190 0190 RR
LINK MAINT 019D 019D RR
LINK MAINT 019E 019E RR
LINK TCPMAINT 0592 0592 RR
```

```
# Check for the existence of a user named LINDFLT:
```

```
dirm for lindflt get lock
```

```
# If this command succeeds, the user exists and must be replaced using this command:
```

```
dirm for lindflt replace
```

```
# Otherwise, issue this command:
```

```
dirm add lindflt
```

---

## 5.28 Create the MASTER z/VM user ID – SL10MSTR DIRECT

Create a file called SL10MSTR DIRECT A with content similar to the sample file provided in Example 5-26, keeping these notes in mind:

- ▶ NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC must correspond to what you specified in MAPSRV and MAPAUTH DIRECT files.
- ▶ Replace AMDISK 0191 3390 AUTOV 5 V64M01 MW with values that correspond to your installation.
- ▶ Our MASTER Linux had an entire mod 9 (0x73AA) available for installing the OS. Make sure that this statement corresponds to the address of your master and its size.

*Example 5-26 Creating MASTER user ID via SL10MSTR DIRECT*

---

```
USER SL10MSTR PASSWORD 1024M 1024M G 64
INCLUDE IBMDFLT
CPU 00 NODEDICATE
CPU 01 NODEDICATE
IPL CMS
MACHINE ESA 4
OPTION QUICKDSP APPLMON
NICDEF 4420 TYPE QDIO LAN SYSTEM CLOUDSWC
AMDISK 0191 3390 AUTOV 5 V64M01 MW
MDISK 0201 3390 1 10016 0X73AA MR ALL WRITE MULTI
```

```
# Check for the existence of a user named SL10MSTR:
```

```
dirm for SL10MSTR get lock
```

```
# If this command succeeds, the user exists and must be replaced using this command:
```

```
dirm for SL10MSTR replace
```

```
# Otherwise, issue this command:
```

```
dirm add SL10MSTR
```

---

## 5.29 Install IBM-System-z.MASTER-1.0-1.s390xr on the MASTER

Copy IBM-System-z.MASTER-1.0-1.s390x.rpm from the Tivoli Service Automation Manager management server to the Linux master ID and install it with the rpm command as shown in Example 5-27.

*Example 5-27 Installing Tivoli Service Automation Manager on Linux MASTER*

---

```
rpm -ivh IBM-System-z.MASTER-1.0-1.s390x.rpm
```

# You must also make sure that the following packages are installed:

```
perl-XML-DOM
perl-HTML-Parser
perl-HTML-Tagset
perl-XML-Generator
perl-XML-RegExp
perl-XML-Writer
perl-libwww-perl
perl-HTML-Tagset
```

# If one or more of these packages are not installed, they must be installed now, using  
# YAST or whatever other package installation method you prefer.

---

## 5.30 Establish appropriate RACF settings

The last part of preparing the z/VM environment involves issuing RACF commands to define RACF resources and to allow and restrict access to those resources. Each set of RACF commands provided in Example 5-28 is preceded by a comment that provides a brief summary of what the commands accomplish. You might need to customize the actual commands based on your environment.

*Example 5-28 Establish appropriate RACF settings*

---

# Login to the MAINT user ID

```
LOGIN MAINT
```

# Permit access to system resources.

# Grant access to the system reader for DATAMOVE, TCPIP and FTPSERVE.

```
RAC PERMIT MAINT CLASS(VMRDR) ID(DATAMOVE) ACC(UPDATE)
RAC PERMIT OPERATOR CLASS(VMRDR) ID(TCPIP) ACC(UPDATE)
RAC PERMIT FTPSERVE CLASS(VMRDR) ID(FTPSERVE) ACC(CONTROL)
```

# Permit access to system resources.

# Permit VSMSEIVE to access the z/VM parm disk.

```
RAC SETROPTS GENERIC(VMMDISK)
RAC PERMIT MAINT.CF1 ACC(ALTER) ID(VSMSEIVE)
RAC PERMIT MAINT.CF2 ACC(ALTER) ID(VSMSEIVE)
```



```

# Configure networking with RACF.
# Define RACF resources for the VSWITCH and MAPLAN.
RAC RDEFINE VMLAN SYSTEM.CLOUDSWC UACC(NONE)
RAC RDEFINE VMLAN SYSTEM.MAPLAN UACC(NONE)

# If your system implements a VLAN, define a RACF resource for it using a RAC DEFINE
# as follows: replace [VLAN] with the numerical identifier for your VLAN. It must be 4 digits,
# so add leading zeroes if necessary.
RAC RDEFINE VMLAN SYSTEM.CLOUDSWC.[VLAN] UACC(NONE)

# Reset all VMLAN definitions.
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) RESET(ALL)
RAC PERMIT SYSTEM.MAPLAN CLASS(VMLAN) RESET(ALL)

# Allow update access to MAINT and DTCVSW1.
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) ID(MAINT) ACCESS(UPDATE)
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) ID (DTCVSW1) ACCESS(UPDATE)

# Allow MAPSRV and TCPIP to connect to the VSWITCH, omitting [VLAN] if your system
# does not implement one.
RAC PERMIT SYSTEM.CLOUDSWC CLASS(VMLAN) ID(MAPSRV) ACCESS(UPDATE)
RAC PERMIT SYSTEM.CLOUDSWC.[VLAN] CLASS(VMLAN) ID(MAPSRV) ACCESS(UPDATE)

# Activate the VMLAN class.
RAC SETROPTS CLASSACT(VMLAN)

# Configure DIRMAINT, DATAMOVE and VSMSEVERE in RACF.
# Give DIRMAINT and DATAMOVE RACF admin authority.
RAC ALU DIRMAINT SPECIAL
RAC ALU DATAMOVE OPERATIONS

# Allow VSMSEVERE to perform password validation.
RAC RDEFINE VMCMD DIAG0A0.VALIDATE UACC(NONE)
RAC PERMIT DIAG0A0.VALIDATE CLASS(VMCMD) ID(VSMSEVERE) ACCESS(READ)
RAC SETROPTS CLASSACT(VMCMD)

# Allow VSMSEVERE to connect to the RACF service machine.
RAC RDEFINE FACILITY ICHCONN UACC(NONE)
RAC PERMIT ICHCONN CLASS(FACILITY) ID(VSMSEVERE) ACCESS(UPDATE)
RAC SETROPTS CLASSACT(FACILITY)

# If protection for DIAG0A0 is not currently active, activate it by issuing.
RALTER VMXEVENT EVENTS1 DELMEM(DIAG0A0/NOCTL)
SETEVENT REFRESH EVENTS1

```

```
# DIAG0A0 is active by default. However, this setting can be changed in the
# currently active VMXEVENT profile by issuing:
RDEFINE VMXEVENT EVENTS1 ADDMEM(DIAG0A0/NOCTL)
```

---

## 5.31 RACF configuration – edit DTCPARMS for VSMERVE and DIRMAINT

Logon to VSMERVE and IPL CMS. Browse (or XEDIT) the file VSMERVE DTCPARMS and verify that the statements shown in Example 5-29 are present. If they are not present, add them.

*Example 5-29 Editing DTCPARMS for VSMERVE and DIRMAINT*

---

```
:Nick.VSMERVE           :Type.server :Class.VSMAPI
                        :ESM_ENABLE.YES
                        :PARMS.-E
                        :Owner.VSMERVE
                        :Exit.VSMEXIT
Nick.VSMAPI             :Type.class
                        :Name.Virtual System Management API server
                        :Command.DMSVSMAS
                        :Runtime.C
                        :Diskwarn.YES
                        :ESM_Validate.RPIVAL
                        :ESM_Racroute.RPIUCMS
```

---

## 5.32 Add RACF commands CONFIGA DATADVH

Logon as MAINT and use **dirm send configxx datadvh** to retrieve the configuration information. Verify that the file includes the set of statements in Example 5-30. After you modify the file to match this example, send the file back to DIRMAINT for activation as described at the end of the example.

*Example 5-30 Adding RACF commands to CONFIGxx DATADVH*

---

```
# While logged in as MAINT, retrieve the CONFIGxx file.

dirm send configxx datadvh
RECEIVE xxx = = A

# Confirm that the file contains these commands, adding lines as needed, then save it.
POSIX_CHANGE_NOTIFICATION_EXIT= DVHXPESM EXEC
LOGONBY_CHANGE_NOTIFICATION_EXIT= DVHXLB EXEC
USER_CHANGE_NOTIFICATION_EXIT= DVHXUN EXEC
DASD_OWNERSHIP_NOTIFICATION_EXIT= DVHXDN EXEC
PASSWORD_CHANGE_NOTIFICATION_EXIT= DVHXPEN EXEC
RACF_ADDUSER_DEFAULTS= UACC(NONE)
```

```

RACF_RDEFINE_VMMDISK_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMPOIX_POSIXOPT.QUERYDB= UACC(READ)
RACF_RDEFINE_VMPOIX_POSIXOPT.SETIDS= UACC(NONE)
RACF_RDEFINE_SURROGAT_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMBATCH_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMRDR_DEFAULTS= UACC(NONE) AUDIT(FAILURES(READ))
RACF_RDEFINE_VMMDISK_DEFAULTS= UACC(NONE)
AUDIT(FAILURES(READ))RACF_VMBATCH_DEFAULT_MACHINES= BATCH1 BATCH2 TREAT_RAC_RC.4=
0 | 4 | 30

```

# If you made changes to the file, “file” it using DIRMAINT with this command:

```

dirm file configxx datadvh
dirm rldcode
dirm rlddata
dirm rldextn

```

---

## 5.33 Summary

The z/VM related configuration steps are now complete. Next, you must customize the DCM in the Managed From environment (Tivoli Service Automation Manager) to enable provisioning using this particular z/VM Managed Through/Managed To environment.





# A

## Appendix

This appendix contains the following:

- ▶ Example of DCM/ XML specifications
- ▶ A list of problems we encountered and fixes or workarounds that we discovered
- ▶ The text of a script that can be used to stop and start the Tivoli Service Automation Manager
- ▶ Notes on capacity planning and z/VM overcommit

## A.1 Example of DCM/ XML specifications

Example 5-31 shows the DCM/XML specifications that we used for xmlimport.

*Example 5-31 Sample DCM/XML specification*

```
<?xml version="1.0" encoding="utf-8"?>
<!--
*****
* Licensed Materials - Property of IBM
* 5724-W78
* (C) Copyright IBM Corp. 2009
* All Rights Reserved
* US Government Users Restricted Rights - Use, duplication or
* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
*****
-->
<!DOCTYPE datacenter SYSTEM "file:../xml/xmlimport.dtd">
<datacenter>
  <kanaha-config>
    <dcm-object id="0" type="managedSystem" name="KANAHA">
      <property component="KANAHA" name="Cloud.DISABLE_HOST_ON_FAILURE" value="false" />
      <property component="KANAHA" name="Cloud.DUAL_NIC_MODE" value="false" />
      <property component="KANAHA" name="Cloud.FAILURE_RETRY_COUNT" value="1" />
      <property component="KANAHA" name="Cloud.HOST_TRANSACTION_NUMBER_LIMIT" value="30" />
      <property component="KANAHA" name="Cloud.IMAGE_FILE_REPOSITORY_ID" value="VMwareFileRepository" />
    /><!--this is required even though we are not using VMWARE (this id is a DCM name) -->
      <property component="KANAHA" name="Cloud.MAX_CONCURRENT_SERVER_OPERATIONS" value="10" />
      <property component="KANAHA" name="Cloud.MPIO_SAN_ENABLE" value="true" />
      <property component="KANAHA" name="Cloud.MhzPerCPUUnit" value="1000" />
      <property component="KANAHA" name="Cloud.TFTP_SERVER_NAME" value="cloud-tftp" />
      <property component="KANAHA" name="Cloud.XEN_HOST_MASTER_IMAGE" value="xen_host_rhel53_64" />
      <property component="KANAHA" name="Cloud.XEN_HOST_POOL_NAME" value="Xen Cloud Pool" />
      <property component="KANAHA" name="Cloud.vmGuestPrefix" value="vm" />
      <property component="KANAHA" name="Cloud.NativeVLANID" value="4" />
      <property component="KANAHA" name="Cloud.ENABLE_VLAN_SUPPORT" value="false" />
      <property component="KANAHA" name="Cloud.xenIPTempRangeStart" value="" />
      <property component="KANAHA" name="Cloud.xenIPTempRangeEnd" value="" />
      <property component="KANAHA" name="Cloud.DYNAMIC_HOST_NETWORKS" value="false" /></dcm-object>
    </kanaha-config>
    <subnetwork address-space="DEFAULT" name="Managed From" ipaddress="9.56.181.128"
netmask="255.255.255.128">
      <blocked-range from="9.56.181.0" to="9.56.181.255" />
      <property component="KANAHA" name="broadcast" value="9.56.181.1" />
      <property component="KANAHA" name="gateway" value="9.56.181.129" />
      <property component="KANAHA" name="globalReserved" value="true" />
      <property component="KANAHA" name="vm-mgmt-route" value="false" />
    </subnetwork>
    <file-repository name="VMwareFileRepository" is-device-model="Cloud File Repository"
root-path="image_repo_disk">
      <nic name="Nic1" failed="false" managed="false" management="false" netboot-enabled="false">
        <network-interface name="nic0" failed="false" managed="false" management="true"
dynamic-ipaddress="false" ipaddress="9.56.181.142" netmask="255.255.255.128"
address-space="DEFAULT" allocation="none" />
      </nic>
```

```

    <property component="KANAHA" name="physicalLocation" value="99999" />
    <property component="KANAHA" name="virtualcenterid" value="99999" />
</file-repository>
<server name="tempNicServer" locale="en_US" ignored-by-resource-broker="false" failed="false">
    <resource name="computer" resource-type="generic" managed="false" partitionable="false">
        <property component="KANAHA" name="computer.manufacturer" />
        <property component="KANAHA" name="computer.serialNumber" />
        <property component="KANAHA" name="computer.type" />
    </resource>
</server>
<subnetwork name="Managed To" ipaddress="129.40.178.0" netmask="255.255.255.0">
    <blocked-range from="129.40.178.1" to="129.40.178.240" />
    <blocked-range from="129.40.178.245" to="129.40.178.245" />
    <blocked-range from="129.40.178.247" to="129.40.178.247" />
    <blocked-range from="129.40.178.249" to="129.40.178.249" />
    <blocked-range from="129.40.178.250" to="129.40.178.255" />
    <property component="KANAHA" name="gateway" value="129.40.178.254" />
    <property component="KANAHA" name="domainname" value="pbm.ihost.com" />
    <property component="KANAHA" name="pridns" value="129.40.106.1" />
    <property component="KANAHA" name="secdns" value="129.40.106.1" />
</subnetwork>
<switch name="CLOUDSWC" locale="en_US" failed="false" number-of-ports="16">
    <property component="KANAHA" name="device-model" value="zVM Virtual Switch" />
    <property component="KANAHA" name="host-platform" value="MAPSRV" />
</switch>
<virtual-server-template name="POK SLES 10 Default VST">
    <resource-requirement resource-type="platform" how-many="0" size="0.0" is-shared="true" />
    <resource-requirement resource-type="cpu" how-many="2" size="0.0" is-shared="true">
        <property component="KANAHA" name="cpu.family" value="s390" />
    </resource-requirement>
    <resource-requirement resource-type="memory" how-many="1024" size="0.0" is-shared="true" />
    <resource-requirement resource-type="nic" how-many="1" size="0.0" group-name="CLOUDSWC"
is-shared="true">
        <property component="KANAHA" name="zVM_NetworkDeviceRange" value="3" />
        <property component="KANAHA" name="zVM_NetworkPortDeviceNumber" value="4420" />
        <property component="KANAHA" name="zVM_NetworkPortType" value="VSwitch" />
    </resource-requirement>
    <resource-requirement resource-type="disk" how-many="0" size="6.9" group-name="vSCSI"
is-shared="false">
        <property component="KANAHA" name="zVM_DiskDeviceNumber" value="0201" />
        <property component="KANAHA" name="zVM_DiskType" value="Minidisk" />
    </resource-requirement>
    <property component="KANAHA" name="host-platform-type" value="zVM" />
    <property component="KANAHA" name="zVM_Prototype" value="LINUX" />
    <property component="KANAHA" name="zVM_Userid" value="" />
</virtual-server-template>
<boot-server name="MAPSRV-bootserver" locale="en_US" is-device-model="zVM_BootServer"
type="zVM" failed="false">
    <property component="KANAHA" name="Hostplatform" value="MAPSRV" />
</boot-server>
<software-stack name="POK SLES10" locale="en_US" version="N/A" stack-type="Declared"
is-device-model="Cloud Suse Linux Operating System">
    <property component="KANAHA" name="soaonramp_vst" value="5464" />
    <property component="KANAHA" name="swType" value="OS" />
    <software-capability type="OS" name="os.family" value="Linux" />

```

```

    <software-capability type="OS" name="os.distribution" value="SLES10 s390x" />
    <software-capability type="OS" name="os.name" value="SLES10 for IBM S/390 and IBM zSeries" />
    <software-capability type="OS" name="os.version" value="10" />
    <software-capability type="OS" name="os.servicepack" value="SP2" />
    <software-requirement name="cpu.family" type="HARDWARE" enforcement="MANDATORY"
hosting="false" accept-non-existing="false">
    <software-requirement-value value="s390" />
</software-requirement>
    <software-requirement name="cpu.type" type="HARDWARE" enforcement="MANDATORY"
hosting="false" accept-non-existing="false">
    <software-requirement-value value="64-bit" />
</software-requirement>
    <software-resource-template name="SLES10-S390X-SP2" software-resource-type="INSTALLATION"
multiplicity-type="N" software-configuration-type="Regular" is-selected="true"
is-default="false" is-deployable="true">
    <template-param name="zVM_Password" value="tld354" parameter-type="String"
multiplicity-type="One" is-hidden="false" is-changeable="true"
is-encrypted="false"></template-param>
    <template-param name="zLinux_RootPassword" value="tld354" parameter-type="String"
multiplicity-type="One" is-hidden="false" is-changeable="true"
is-encrypted="false"></template-param>
</software-resource-template>
</software-stack>
    <image name="POK SLES10 SP2 Minidisk" image-type="Golden_Master" description="Prepared for
TSAM" locale="en_US" version="1.0" boot-server="MAPSRV-bootserver" status="tested"
is-device-model="SOAonRAMPimage" software-module="POK SLES10" priority="1">
    <property component="KANAHA" name="zVM_CloneDisks" value="201" />
    <property component="KANAHA" name="zVM_DiskOwnerId" value="SL10MSTR" />
    <property component="KANAHA" name="zVM_Prototype" value="LINUX" />
    <property component="KANAHA" name="zVM_SystemDisk" value="201" />
</image>
    <spare-pool name="System z pool">
    <server name="MAPSRV" locale="en_US" is-device-model="SOAonRAMP_HostPlatform"
ignored-by-resource-broker="false" failed="false">
    <network-interface failed="false" managed="false" management="true"
dynamic-ipaddress="false" ipaddress="129.40.178.250" netmask="255.255.255.0"
address-space="DEFAULT" allocation="none" />
    <nic name="NIC-ConnectedTo-CloudSWC" locale="en_US" failed="false" managed="true"
management="true" netboot-enabled="false" group-name="CLOUDSWC" is-vlan-aware="false">
    <property component="KANAHA" name="zVM_NetworkPortType" value="VSwitch"
description="Type of NIC connection - VSwitch" />
    <property component="KANAHA" name="zVM_NetworkVSLanName" value="CLOUDSWC"
description="VSwitch name" />
</nic>
    <property component="KANAHA" name="NIC-ConnectedTo-CloudSWC" value="CLOUDSWC" />
    <property component="KANAHA" name="MaxDiskSize" value="6.9" />
    <property component="DEPLOYMENT_ENGINE" name="LPAR.cec_type" value="zVM" />
    <property component="DEPLOYMENT_ENGINE" name="host-platform-type" value="zVM" />
    <property component="KANAHA" name="physicalLocation" />
    <property component="DEPLOYMENT_ENGINE" name="vmsmserve-address" value="172.16.0.1"
description="Guest Lan IP address for MAP to contact VSMSEVER on port 845 with MAPAUTH's userid
and password" />
    <property component="DEPLOYMENT_ENGINE" name="vmsmserve-port" value="845" />

```



```

    <sap name="PING" is-device-model="ICMP Ping Service Access Points" locale="en_US"
protocol-type="ipv4" app-protocol="ICMP" context="NOCONTEXT" port="0" auth-compulsory="true"
role="host">
    <default-sap operation-type="ping" />
    <credentials search-key="master" is-default="true">
        <password-credentials username="root" password="tld354" is-encrypted="false" />
    </credentials>
</sap>
    <sap name="zVM SSH" is-device-model="SSH Service Access Point" locale="en_US"
protocol-type="ipv4" app-protocol="SSH" context="NOCONTEXT" port="22" auth-compulsory="true"
role="host">
    <default-sap operation-type="execute-command" />
    <credentials search-key="master" is-default="true">
        <password-credentials username="root" password="someword" is-encrypted="false" />
    </credentials>
</sap>
    <sap name="SMAPI" locale="en_US" protocol-type="ipv4" app-protocol="LOCAL-EXEC"
context="NOCONTEXT" port="845" auth-compulsory="true" role="host">
    <credentials search-key="master" is-default="true">
        <password-credentials username="MAPAUTH" password="tld354pw" is-encrypted="false" />
    </credentials>
</sap>
<host-platform />
<resource name="Platform" resource-type="platform" managed="true" partitionable="false">
    <property component="KANAHA" name="platform.architecture" value="390" />
</resource>
<resource name="CPU" resource-type="cpu" managed="true" partitionable="true">
    <property component="KANAHA" name="cpu.family" value="s390" />
    <property component="KANAHA" name="cpu.size" value="2" />
    <property component="KANAHA" name="cpu.type" value="64-bit" />
</resource>
<resource name="Mem" resource-type="memory" managed="true" partitionable="true">
    <property component="KANAHA" name="memory.size" value="4096" />
</resource>
<resource name="POOL0" resource-type="disk" group-name="vSCSI" managed="true"
partitionable="true">
    <property component="KANAHA" name="disk.size" value="35" />
</resource>
</server>
<property component="KANAHA" name="cloud" value="true" />
</spare-pool>
<file-repository name="dcx48b" is-device-model="CDSFileRepository" root-path="/export">
    <network-interface failed="false" managed="false" management="true"
dynamic-ipaddress="false" ipaddress="9.56.181.141" netmask="255.255.255.128" allocation="none"
/>
    <use-workflow workflow-id="FileRepository_GetFile_SCP" />
    <sap name="LOOPBACK" locale="en_US" protocol-type="ipv4" app-protocol="UNKNOWN"
context="NOCONTEXT" port="0" auth-compulsory="true" role="host">
    <default-sap operation-type="file-transfer" />
    <default-sap operation-type="execute-command" />
    <default-sap operation-type="ping" />
    <credentials search-key="loopback" is-default="true">
        <password-credentials username="root" password="tld354" is-encrypted="false" />
    </credentials>
</sap>

```

```
<sap name="SCP" locale="en_US" protocol-type="ipv4" app-protocol="SCP" context="NOCONTEXT"
port="22" auth-compulsory="true" role="client">
  <credentials search-key="master" is-default="true">
    <password-credentials username="root" password="t1d354" is-encrypted="false" />
  </credentials>
</sap>
<sap name="SSH" locale="en_US" protocol-type="ipv4" app-protocol="SSH" context="NOCONTEXT"
port="22" auth-compulsory="true" role="client">
  <credentials search-key="master" is-default="true">
    <password-credentials username="root" password="t1d354" is-encrypted="false" />
  </credentials>
</sap>
<access-domain-membership>
  <access-domain-name>sample:all-objects</access-domain-name>
</access-domain-membership>
</file-repository>
</datacenter>
```

---

## A.2 Common configuration errors

We encountered a variety of errors and situations that resulted from the fact that we were essentially discovering the processes and sequence of events required to implement a deployment of a Cloud on System z. This section documents the challenges we encountered, so that you can benefit from our experience and avoid the same problems.

### **COPQLX008E – The system cannot resolve the query**

Message or symptoms:

```
COPQLX008E "The system cannot resolve the query:
/Server[@id=$tempNicServerID]/@id"
```

Cause: This indicated that we had not imported the tempNicServer definition as part of the xmlimport.

Solution: We imported the following snippet of XML and added the same to our XML file. On the next iteration of the building/rebuilding process the updated XML file contained the tempNicServer definition.

```
<server name="tempNicServer" locale="en_US" ignored-by-resource-broker="false"
failed="false">
  <resource name="computer" resource-type="generic" managed="false"
partitionable="false">
    <property component="KANAHA" name="computer.manufacturer" />
    <property component="KANAHA" name="computer.serialNumber" />
    <property component="KANAHA" name="computer.type" />
  </resource>
</server>
```

### **COPCOM231E – The system cannot find the zVM\_Userid property**

Message or symptoms:

```
COPCOM231E The system cannot find the zVM_Userid property.
```

Cause: This message can result if:

- ▶ After importing the XML, you forget to set value for soaonramp\_vst to be equal to the ObjectID of the Virtual Server Template
- ▶ The Virtual Server Template does not have the variable defined.

Solution: We located the ObjectID of the VST and set the set the soaonramp\_vst after the xmlimport.

### **COPDEX123E – A MissingInputParameter exception occurred**

Message or symptoms:

```
COPDEX123E A MissingInputParameter exception occurred.
The exception was caused by the following problem:
CTJZH7000E + domainSuffix + .
```

Cause: We did not have a domain assigned to the subnet (note the bold statement in the following code, which was missing from our imported XML).

```
<subnet name="zBMC Pok" ipaddress="129.40.178.0" netmask="255.255.255.0" >
  <blocked-range from="129.40.178.1" to="129.40.178.240" />
  <blocked-range from="129.40.178.245" to="129.40.178.245" />
```

```
<blocked-range from="129.40.178.247" to="129.40.178.247" />
<blocked-range from="129.40.178.249" to="129.40.178.249" />
<blocked-range from="129.40.178.250" to="129.40.178.255" />
<property component="KANAHA" name="gateway" value="129.40.178.254" />
<property component="KANAHA" name="domainname" value="pbm.ihost.com" />
</subnetwork>
```

Solution: We manually updated the DCM using the Maximo interface and also added the following to our XML so it would be complete on the next xmlimport:

```
<property component="KANAHA" name="domainname" value="pbm.ihost.com" />
```

## **error UNKNOWN\_ERROR 596 3618**

Message or symptoms:

error UNKNOWN\_ERROR 596 3618

This error was received in combination with this message:

CTJZH7506E The RPC client returned return code /op

Solution: We received this message when a guest we were trying to provision already existed. This occurred because our deprovisioning process was not working. If you encounter this situation, you can eliminate the old or offending previously defined guest by logging in as MAINT and issuing this DIRM command for each user ID that you need to deprovision:

DIRM FOR userID PURGE CLEAN

**Important:** Before doing this, make sure that no one is logged as userID.

## **COPDEX123E – A WorkflowException exception occurred**

Message or symptoms:

COPDEX123E A WorkflowException exception occurred. The exception was caused by the following problem:

CTJZH7031E + COPDEX172E The route variable is a single value variable, cannot assign an array value to it.

Cause: The vm-mgmt-route variable connected to the Cloud Management VLAN (9.56.181.128) was set to true. We do not allow routes back to the Tivoli Service Automation Manager machine.

```
<subnetwork address-space="DEFAULT" name="Cloud Management VLAN" ipaddress="9.56.181.128"
netmask="255.255.255.128"><blocked-range from="9.56.181.0" to="9.56.181.255"/><property
component="KANAHA" name="broadcast" value="9.56.181.1"/>
<property component="KANAHA" name="gateway" value="9.56.181.129"/>
<property component="KANAHA" name="globalReserved" value="true"/>
<property component="KANAHA" name="vlan_id" value="4"/>
<property component="KANAHA" name="vm-mgmt-route" value="true"/>
</subnetwork>
```

Solution: We changed the XML to indicate value="false" as follows:

```
<property component="KANAHA" name="vm-mgmt-route" value="false"/>
```

## **COPDEX123E – A MissingInputParameter exception occurred**

Message or symptoms:

COPDEX123E A MissingInputParameter exception occurred. The exception was caused by the following problem:

CTJZH7000E + primaryDNS

Cause: The subnetwork for the provisioned servers was missing parameters (shown as bold in the following code).

```
<subnetwork name="zBMC Pok" ipaddress="129.40.178.0" netmask="255.255.255.0" >
<blocked-range from="129.40.178.1" to="129.40.178.240" />
<blocked-range from="129.40.178.245" to="129.40.178.245" />
<blocked-range from="129.40.178.247" to="129.40.178.247" />
<blocked-range from="129.40.178.249" to="129.40.178.249" />
<blocked-range from="129.40.178.250" to="129.40.178.255" />
<property component="KANAHA" name="gateway" value="129.40.178.254" />
<property component="KANAHA" name="domainname" value="pbm.ihost.com" />
<property component="KANAHA" name="pridns" value ="129.40.106.1"/>
<property component="KANAHA" name="secdns" value ="129.40.106.1"/>
</subnetwork>
```

Solution: We added the pridns and secdns property XML statements [shown in bold].

## **COPDEX123E – A MissingInputParameter exception occurred**

Message or symptoms:

COPDEX123E A MissingInputParameter exception occurred. The exception was caused by the following problem:

CTJZH7517E\_INSUFFICIENT\_TARGET\_DISK\_SIZE + 0201 + 0201 +

Solution: In this case, the image the was registered with only 1 GB of disk space. We edited the DCM object for that registered image virtual server template (Minimum configuration for SLES10 and Recommend configuration for SLES10) so that the required disk space was 7 GB.

## **COPDEX137E**

Message or symptoms:

COPDEX137E There is no provisioning workflow that implements the OperatingSystem.SetAdminPassword' logical management operation associated with object 'SLES10'. The software stack didn't have the right device model.

Solution: We added the is-device-model as follows:

```
<software-stack name="POK SLES10" locale="en_US" version="N/A" stack-
type="Declared" is-device-model="Cloud Suse Linux Operating System"
```

## **COPDEX123E**

Message or symptoms:

COPDEX123E A timeoutError exception occurred. The exception was caused by the following problem: ERROR: Device not responding or unexpected response.

Cause: This error was encountered during an SCP file copy. The device was a SLES 11 Linux machine.

Solution: We determined that the problem was the bash prompt having color in it (the terminal codes were not read correctly) The fix was to set the PS1 env variable in /root/.bashrc to  
export PS1='\h:\w # '

## **CTJZH7506E\_UNKNOWN\_ERROR**

Message or symptoms:

CTJZH7506E\_UNKNOWN\_ERROR + 500 + 8 + CTJZH7512E\_CREATE\_NEW\_GUEST\_FAILED

Cause: Dir Maint was not logged on.

**Solution:** We made sure that the DIRMAINT Service machine was logged on.

### **TSAM does not update the provisioned guests' /etc/resolv.conf**

Message or symptoms: Tivoli Service Automation Manager does not update the provisioned guests' /etc/resolv.conf with the DNS specified in the DCM (<subnetwork/> object) and the provisioned guests are left with an exact copy of MASTER's /etc/resolv.conf.

Solution: The MASTER's /etc/resolv.conf must contain the correct nameservers for the provisioned guests.

### **TSAM using auto-generated hostnames for the provisioned guests**

Message or symptoms: The Tivoli Service Automation Manager *managed from* host must be able to do a reverse DNS lookup using the IP addresses you specified in the DCM. This is the method by which the provisioned guests' hostnames are determined. You must register each of the potentially provisioned guests' IP addresses and hostnames with your nameserver. If the lookup fails, either because the correct nameserver is not listed in /etc/resolv.conf, or because the IP addresses and hostnames are not registered with the nameserver, the correct hostnames will not be retrieved. Instead, an alphanumeric string, the hexadecimal representation of the IP address, will be used for each of the guests' hostnames.

Note that we were unable to register the IP addresses that were available to provisioned guests with our site's DNS server, so we installed **dnsmasq** (which provides DNS) on our local subnet and entered the information there.

If you use dnsmasq or a similar method to store the hostnames of the provisioned servers, then the server it is installed on must be the first nameserver listed in /etc/resolv.conf. If not, Tivoli Service Automation Manager might not be able to look up the hostnames. You can test this by issuing **dig +x <IP address>**, which should return the correct host name.

Solution: The hostnames and IP addresses for each address available for provisioning must be registered on the first nameserver listed in /etc/resolv.conf on the management server.

### **VSWITCH Layer 2 versus Layer 3**

Message or symptoms: A provisioning request fails because Tivoli Service Automation Manager attempts to attach a virtual NIC as a layer 3 device to a layer 2 VSWITCH.

By default, Tivoli Service Automation Manager will have the provisioned guests attempt to attach virtual NICs to the z/VM VSWITCH as though it was configured as a layer 3 switch (the IP keyword was specified in the DEFINE VSWITCH command). If your VSWITCH is actually configured as a layer 2 switch, (the ETHERNET keyword was specified on the DEFINE VSWITCH command) this step will fail silently. This will cause the entire provisioning process to fail.

Solution: This is really a workaround, not a solution. The developers have addressed the issue and the fix will be available in a future release of Tivoli Service Automation Manager. The workaround is to edit the file `personalize_SLES10.sh` in the `/opt/IBM/AE/AS/` directory on MASTER. Change `"QETH_LAYER2_SUPPORT='0'"` to `"QETH_LAYER2_SUPPORT='1'"` on line 177 and save the file. Then run the provisioning workflow again.

## A.3 Stopping and starting Tivoli Service Automation Manager

We used the script in Example A-1 to stop and start the Tivoli Service Automation Manager.

*Example A-1: Script to stop and start TSAM*

---

```
### BEGIN INIT INFO
# Provides: TSAM
# Required-Start: cron
# Required-Stop:
# Default-Start: 2 3 5
# Default-Stop: 0 1 6
# Description: Start or stop DB2 instance, LDAP server, HTTP Server, Deployment
Manager, Agent Manager, nodeagent and MXServer
### END INIT INFO
DB2ITDSIHS_SCRIPT="/root/bin/db2itdsihs.sh"
test -f $DB2ITDSIHS_SCRIPT || exit -1
TIO_SCRIPT="/opt/IBM/tivoli/tpm/tools/tio.sh"
test -f $TIO_SCRIPT || exit -1
if [ -f /etc/rc.status ]; then
. /etc/rc.status
rc_reset
fi
case "$1" in
start)
echo -e "Starting DB2 instance, LDAP and HTTP servers"
$DB2ITDSIHS_SCRIPT start
echo -e "Starting TPM engines"
su - tioadmin -c "$TIO_SCRIPT start"
test -f /etc/rc.status && rc_status -v
;;
stop)
echo -e "Stopping TPM engines"
su - tioadmin -c "$TIO_SCRIPT stop wasadmin password"
echo -e "Stopping DB2 instance, LDAP and HTTP servers"
$DB2ITDSIHS_SCRIPT stop
test -f /etc/rc.status && rc_status -v
;;
status)
echo -e "Checking TSAM status"
test -f /etc/rc.status && rc_status -v
;;
*)
#if no parameters are given, print which are available.
echo "Usage: $0 {start|stop|status}"
exit 1
;;
esac
exit 0
```

---

## A.4 Capacity planning and z/VM overcommit

It is recommended that you include capacity planning and z/VM overcommit as part of your overall process of implementing a cloud on z/VM to provision Linux. Capacity planning is an important consideration when dealing with cloud computing, especially in the context of the power of virtualization of resources provided by z/VM.

For instance, depending on the workload that a service requires, CPU can be overcommitted by a factor of 1 to 10. This means that for every real CPU allocated to the z/VM LPAR, 10 can be utilized for this service. On other hand, memory overcommit is somewhat different from CPU overcommit. Generally speaking, experience shows that memory can be committed by a factor of 1 to 3. For every real GB of memory, 3 GB can be used for services.



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this paper.

## IBM Redbooks documents

The following IBM Redbooks publication provides additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *Provisioning Linux on IBM System z with Tivoli Service Automation Manager*, REDP-4663

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks publications, at this website:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Other publications

This publications are also relevant as further information sources:

- *Tivoli Service Automation Manager V7.2 Installation and Administration Guide*, SC34-2565

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)







# Deploying a Cloud on IBM System z



## Understanding cloud component models

## Configuring z/VM and Tivoli Service Automation Manager

## Deploying the cloud with a step-by-step checklist

Cloud computing, using shared resources in public spaces instead of in-house IT organizations, is the latest thing in IT. Lines of business even bypass their own IT shops to take advantage of external providers of cloud offerings. However, many of the users that employ public cloud services have not considered issues involving security, compliance, and availability.

Cloud represents a new business model that requires a process discipline as well as the use of a corresponding set of technologies. The new model requires an understanding of the hardware configuration, software images, a virtualized storage infrastructure, and network management.

For many organizations that have mainframe resources, the IT professionals already manage these different disciplines and aspects of resources as part of their overall management of the platform. The mainframe's proven capability to efficiently and securely provide virtualization, combined with the existing skills in the IT organization, suggest that in-house mainframe resources provide an ideal environment in which to pilot cloud computing.

This IBM Redpaper document describes the steps we took to create an environment that can efficiently deploy and manage a cloud in a Linux-based Infrastructure as a Service (IaaS).

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

## BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)